Data Dictionaries: Bane or Boon

Stephen M. Butler
PROBUS International, Inc.
8815 - 106th St. E.
Puyallup, WA 98373

Introduction

Data Dictionaries (DD) have been around for over a decade. Even so, questions still arise: What is a DD? What are the benefits? Are these benefits real? The answers are varied and complex. The bottom line rests with the particular user and the application. This paper will point to the anticipated bottom line and give the circumstances under which the benefits should be realized.

DDs (Data Dictionaries) are the repository of data about data -- in short, meta-data. In similar fashion a Payroll system is the repository of data about employees and Accounts Payable the repository of vendors who need to be paid. Just as the Payroll and Accounts Payable data is important to the future development of a company, so is its meta-data.

The DD provides the mechanism to ensure that the benefits of database systems are fully realized. Just as a database reduces or eliminates redundancy of data (vendor name in six separate files) so the DD eliminates multiple definitions of the same item or, worse yet, distinct pieces of data defined as the same.

For example, in 1987 a major company in the Pacific Northwest was installing a commercial package on one of it's systems. This package made use of several databases. Ad hoc reports were needed that could quickly be developed using a report writer. The first step involved loading the database definitions into a DD. During this step several conflicting definitions for data names were discovered. In some places the Item Number was 6 characters while others had it as 10 characters. Worse yet was the discovery that these were indeed unrelated pieces of data. Different definitions were given the same name.

Consistency of data definition is very crucial for the successful implementation of 4th G/L and Relational DB technologies. It is also paramount to the survival of a company in this information age.

In fact, DD along with the DBA (Data Base Administrator) were projected to be the premier organization of the '90s. This combination was to provide the company with the knowledge of what information the company

had available. With this information the company could chart its way through the application backlog using the leading edge of technology. The battle cry could have been "Corporate VP or BUST"!

BOON CITY

Within the HP-3000 community the early 1980s saw many vendors head toward the mecca of DD. Even Hewlett-Packard bought one and named it, appropriately, DICTIONARY/3000. The band wagon appeared all set to move. The key word became PROTO-TYPING. The tools were 4TH G/L and, of course, DD.

The benefit list became long (and sometimes emotional):

- common data definitions
- central repository of meta-data
- run time data resolution
- proto-typing
- automated documentation
- user views (ie, logical data design vs physical structure)
- language integration
- report writer generation
- data base definition
- etc.

If the above list leaves the impression that 4th G/Ls were the issue, than one understands where DD stood in terms of understanding. So, what were/are the benefits of, say, DICTIONARY/300Q.

-- IMAGE Schema Generations

A project within Weyerhaeuser company kept the IMAGE schemas inside the DD. This was much easier than keeping separate schema files up to date. Modifications rarely caused problems (such as syntax errors or unreferenced data sets).

-- Ad Hoc Report Generators

A by product of this came in the form of INFORM as an ad hoc report generator. The data definitions were already in the DD. Some collections of data were brought together in INFORM GROUPS. A training program was instituted for the end users (several sites). This program stressed hands on examples using copies of live data. The trainees became familiar with both DICTIONARY/3000 and INFORM.

A recent conversation with the project manager revealed that this combination was very successful and still (some three years later) viable. In fact, this allowed the development staff to address some new major areas that would have been impossible otherwise.

-- Data Documentation

Kitsap County provides the development staff with updated element definitions. In fact, the development staff is expected to add new and/or modify existing definitions with the DD. Several elements exist that have nothing to do with IMAGE databases. These provide linkages for the development staff to locate the appropriate technical information.

-- On-line User Help

In addition, they developed an interface between VIEW/3000 and DICTIONARY/3000 that provides on line help to application users. The field names are entered into the DD along with the help text for the user. This allows them to produce a users manual that is viable across applications. The individual user departments are free to incorporate any specific policies or procedures into the manual. Thus the development staff is spared the burden of providing a totally customized manual. (The "help text" in the DD serves as internal documentation also. A nice double benefit.)

-- Language Support

Several languages now interface with various DDs to provide increased productivity. The authors primary experience has been with the TRANSACT language with DICTIONARY/3000. Various clients have reported program development gains of 8 to 10 times over COBOL. The author prefers a solid conservative 3 to 5 fold gain. Other developers with differing packages (PROTOS, COGNOS, SPEEDWARE, etc.) indicate similar advantages.

Almost every productivity aide in this arena has recognized the need for DD support. It is analogous to writing a single WORKING-STORAGE SECTION for COBOL and using it in all future programs with the compiler removing those portions that are unused.

All of the above made unique uses of the DD. None of them exploited the DDs to their full capability. This author wrote "DICTIONARY/3000 WALK THROUGH" to demystify the inner workings so that greater exploitation would occur. Then followed "DICTIONARY/3000: HUB OF SYSTEMS DEVELOPMENT AND DOCUMENTATION". Suffice to say, the tools necessary to fully utilize the ideas promulgated have not been forthcoming.

In fact, many sites still wonder what possible use they could have for a DD. Those who have found uses have also found frustrations.

-- Data Black Hole

Some sites have been chagrined to find their carefully loaded descriptions do not appear on the expected reports. For example, the DICTIONARY/3000 report of elements and sets within a database prints the ELEMENT to FILE description. The carefully entered ELEMENT descriptions are not available in that format. Conversely, the ELEMENT to FILE descriptions are not available with the ELEMENT listing.

The later makes some sense, but the former begs for an explanation. So far, no one has supplied one that is satisfying.

-- Competing Usage

As stated above, many vendors responded to the siren call and produced their own version of a DD. The major "productivity" language developers have found that a DD is needed. But, each has incompatible functions when compared to others.

The shop that has more than one such aide with the attendant discrepancies has relegated DD to the language support role. This is unfortunate as the major benefits of a DD are lost. Worse yet is the growing perception that a DD is something to be endured while attempting to gain productivity.

Some DDs provide for interfaces into DICTIONARY/3000 (and maybe other products) but the effort is time consuming and frustrating. Attempts to keep the DDs in sync are worse than handling redundant data in pre-database applications.

-- Ignorance is bliss

Any perceived benefits are lost in a shop using standard languages such as COBOL. These ignore the gold mine (some term it a coal shaft) of data definitions and require the development staff to re-type much of the

structure. The available utilities do not provide the configurability needed to support variations between shops.

There are few pre-processors that retrieve data structures from the DD and insert them into a standard language. Those that do require their own specialized DD. This only adds to the growing burden of keeping everything in sync.

-- Time Warp

Perhaps it is better said "user warped"! With the available tools that directly change the database structure, not one will keep a DD in step. As time marches on, the difference between reality and the DD drifts from a hairline crack into a full concussion. The resulting headache is usually fatal for the DD and may even cast a coma over the associated shop.

-- Frozen meta-data

Some shops have found that the DD has provided such help as on-line user manuals. This close integration with the application is greatly appreciated until a change is needed in the on-line help. There is a story of a complete system recompile just to incorporate a grammatical change in the help text.

Even more amazing is the concept of compiled DD. Perhaps the first such encounter is with the IMAGE root file. For obvious reasons, this does need "compilation".

Conversely, the DD is a central library of information that is dynamic. Even as applications and systems change so must the information library. It is not true that changes in the information library (DD) should require change in the application.

-- Friday on strike

The utilities that do exist tend to support the incorporation of information into the DD. This information is usually in a raw form that needs further clarification by the development staff (such as where decimal points really belong in an IMAGE J2 field). Very few utilities travel the opposite path.

The development staff is further stymied by the lack of support during the migration from development to production. This is especially true where there is a phased release going to more than one site. The above comments regarding interaction between DD of competing vendors could be repeated here. The usual method of transfer results in production seeing

everything development had in the DD at release time. The concept of phased releases is foreign to most DDs.

Even further astray is support for distributed DDs. Many might claim that the current multi-vendor solution is distributed. The lack of support within a single vendor's DD makes this situation nearly identical to the multi-vendor bottleneck.

OUT OF THE QUAGMIRE

The above sounds hopeless. While the situation is more dismal than expected, it certainly isn't hopeless. The route from Bane Villa to Boon City need not end in the Quagmire. The mile posts along the way must contain the following.

-- More Useful Utilities

The Data Black Hole concept needs reversed. Documentation and other descriptions that are input must be readily regurgitated by the DD. Such areas of major concern are User On-line Help and User Manuals. The greatest distress for the development staff rests with end user documentation. Any utility that will reduce this burden (notice the word REDUCE) will be a great benefit.

Many shops take the data descriptions from the DD to a flat file and then proceed to rework them into user documentation. An utility that understands user documentation and is configurable via a meta-language in the DD will greatly facilitate in this area.

A similar utility should allow the on-line user direct access to the documentation stored in the DD. Such items as Departmental Policy and Procedures, Screen Level Help, and User Enterable Field documentation could be readily available at a keystroke.

Data validation rules would also be accessible. The human readable version could be generated as needed from an internal control language. This language would allow the development staff to incorporate data validation codes within the application.

In fact, the DD should provide language snippets to go in COPYLIBs or otherwise be included in standard 3rd generation languages (COBOL, FORTRAN, ...). These snippets would be both Working-Storage (COBOL) and procedural code.

One major area of the DD is the documentation of what programs use which elements from identified files (datasets, flat files, etc.). The natural

outgrowth is code written in any language to retrieve and/or update said elements. This need not be a complete program but would be code the programmer could slap into a program and utilize. Quick integration into a program is the key ingredient.

Another key ingredient is internal documentation that the DP auditors love so much. Since the DD already knows the structure of the databases, it should provide the standard chart consisting of triangles and trapezoids. It should also be able to reduce the number of line crossings to a minimum or even eliminate them with unique but reasonable layouts.

Similar utilities would provide program level documentation. Something akin to HIPO charts or high level data flow diagrams should be available.

-- Productivity and Development Aides

There are many areas in which development can be aided. The standard 4th G/L approach has been in the arena of program construction. Other areas are just as critical to the complete life cycle of an application. The initial design phases are usually completely redeveloped in later stages. This initial effort needs to be captured in a manner that is easy to regurgitate in the form needed later.

In fact, by the time detailed system design is started there is a great deal of information known that should be in the DD to facilitate the data structure design. With the added knowledge of data dependencies the DD should provide a normalized (at least 3rd normal form) logical data design. Given a few rules regarding performance considerations, the system could even produce the first cut at a physical database layout. (In the HP-3000 world the distinction between logical and physical design is greatly blurred.)

-- Extendible and Configurable

While the current state of DD art is very far from what has been described so far, it is doubtful that any comprehensive DD solution will be the answer to every need. Therefore the basic structure needs to be extendible by individual sites. The basic implementation should be very rich to negate many needs for the extendible. But, the DD should support a very extensive capability for extension.

The basic structure should be amenable to some configurability. Such things as the ELEMENT NAME size along with other attributes need to be configured by the individual sites.

-- Vendor Integration

The current plethora of vendor specific DD systems is a major roadblock. The DD of the future needs to support varied needs according to other vendor software. In fact, the ideal DD would allow the installation procedures to compare the current DD structure against the needs of the new package. Here again, Extendibility and Configurability come to play.

This is akin to the varied remote control devices for TVs, radios, video decks and satellite dishes now on the market. The solution is one remote control that can learn all the functions of the others combined. The DD of the future must be able to incorporate the needs of other packages.

-- Language Support

Mention has already been made of utilities that would assist the programmer by generation of data structure and procedural code. A further integration is needed whereby the compilers of 3rd G/L software can interrogate the DD to supply field definitions or even round out some sections of procedural code.

This sounds like infringement on the current slate of 4th G/L generators. However, most of these generators are lost in the foursome of ADD, UPDATE, DELETE and REPORT functionality. They do not address the more complex issues of A/R aging, Payroll processing, or General Ledger cycles.

The 4th G/L generators will become capable of more advanced processing. Likewise the DD should become smarter about languages and how to generate code for them. Even so, it is incumbent upon the 3rd G/L compilers to learn about the DD.

-- Programmatic Interfaces

All of the above areas demand smart programmatic interfaces into the DD. These interfaces must be bi-directional. In addition, the inquiring software must be able to discern what extension are in place and how the current site has configured the DD.

This would allow a new package to automatically extend the DD where needed and modify its own code to comply with local configurations where appropriate. Naturally, there will always be areas where the shop will have to choose between alternatives. The choices will be easier when the software lays out the pros and cons.

Much current software could benefit by direct interrogation of the DD. Major examples are the ADAGER and DBGENERAL utilities. Ideally the DD

would drive either or both utilities (configurable option!) to change the database from what was to what is to be (or not to be).

Alternatively, either or both of these products should be able to update the DD to reflect the actual changes made.

TO EPCOT

The reader may conclude that the author has gone completely futuristic. This may be even more apparent when a careful review indicates the need for an EXPERT SYSTEM. Indeed, the Data Dictionary Software must become an EXPERT SYSTEM. It needs to be an expert regarding systems development, design and documentation.

Various interfaces must be provided including the board room interface where questions such as, "If we collect this addition information regarding our clients (or products or ...), will we have a competitive edge in our industry?" Therefore, this system must also become an expert regarding the company and industry it is functioning within.

The programmer interface is much different. Much of the grunt work of coding would be done by the DD system. That would leave the programmer free to concentrate on the more unusual requirements. These requirements would be researched through interaction with the DD to gather available documentation. Missing information would be incorporated by the programmer in accordance with the local shop style.

The user interface becomes subtle. A pause, erroneous input or other indications of confusion would be met by a system ready to analyze the current situation and suggest alternatives. Always would be the fall back where the user would interrogate the DD for information. This may be more detailed, more general, or further afield than initially offered.

To do this, the DD system must expand its own knowledge base. That is the major premise of expert systems. They learn from the information that is incorporated therein. This implies that the DD would learn about changes made to itself and incorporate that knowledge into its expert base.

Not only would this expert system have human interfaces but it would allow inquiries from applications and other systems. This would range from the current set of edit specs for a field to the current configuration of the DD.

CONCLUSION

Bane Villa or Boon City? The current situation is both. The maintenance of element definitions and IMAGE schema files is a snap. Full DICTIONARY/3000 implementation suffers from the Black Hole theory. Multiple vendor shops with the attendant DDs have found the dream of consistency across dictionaries to be elusive.

So what is a shop to do? The simple things:

- Keep the IMAGE structure in the DD.
- Maintain the Element documentation both for technical level and end user help facilities.
- Utilize the ad hoc report generator interfaces.
- Where applicable, utilize the 4th G/L generators.

And, finally, let your vendors know that integration is the future. The "divide and conquer" situation must become "let's hang together or we'll hang separately."

Interestingly, help may be coming from an unexpected source. The AI expert system developers have found relational technology to merge very nicely with knowledge basis. Their solutions for managing this information base are starting to sound like Data Dictionary systems. Soon someone will make the giant leap that intertwines the expert system with the description of its knowledge base. A smart DD is nothing more than an expert system driving the description of itself in order to facilitate its own development with the goal being design, documentation and development of expert applications.