Generating requirements from enterprise analysis

by DAVID V. KERNER
IBM DP Professional Center
San Francisco, California
and
ASHOK MALHOTRA
IBM Research
Yorktown Heights, New York

ABSTRACT

BICS is an enterprise analysis methodology based on a normative, top-down model of the information requirements of a corporation. This paper provides an overview of BICS and introduces REOGEN, an extension of the BICS methodology that generates definitions for the data required to manage the organization and specifications for the processes required to maintain the data.
INTRODUCTION

Most of the activity in the area of requirements for data processing applications has been in the definition of languages for specifying and analyzing requirements. We feel that from an analysis of the goals and policies of an organization it should be possible to generate its data processing requirements. In this paper we discuss REQGEN a system that attempts to do this. REQGEN (requirements generator) is an extension of BICS, an enterprise analysis methodology based on a normative, top-down model of the information requirements of a corporation.

The paper begins by describing BICS and a language called EAS-E that was used to implement the BICS methodology and has heavily influenced the requirements work. It then discusses a BICS study. Finally, in the largest section, it discusses the extension of the methodology for generating requirements.

BICS

BICS (Business Information Control Study), a development of BIAIT, is an enterprise analysis methodology that builds a vehicle for communicating information requirements from the end user to the data processing function. This is accomplished by building an information model of the organization that is based on the options it selects to manage the orders it receives. The methodology and the model have been successfully validated in more than 20 studies all over the world.

Orders, in BICS terms, are requests made from outside the organization that the organization must respond to. The response consists of supplying a product or a service. The product may be something manufactured (such as a CPU), a report, or even a verbal response. For each type of order the organization has certain management options, such as whether to produce to order or to stock, or whether the customer is billed or pays cash.

After the order types have been identified, a set of questions is asked about each order type to determine the management options. The answers to these questions identify the business functions. The business functions are (1) Payment: “Do you bill the customer?” (2) Delivery: “Do you keep records for use in processing the order?” (3) Customer profile: “Do you keep records, by individual customer for planning purposes?” (4) Ownership: “Do you keep title to the ordered product after delivery?” (5) Tracking: “Do you initiate service, change, or recall of a product after it has left your organization?” (6) Specification: “Do you create product/service specifications?” (7) Manufacture: “Do you make the product or service?” (8) Stock finished goods: “Do you stock the ordered product?”

A “Yes” answer to any one of the questions implies a set of data classes that must be present in the organization for the particular order type.

Next the study team takes each order type and determines which organizational units are actually accountable for the data in the data classes associated with that order type. Accountability has four aspects: i.e. accountability for (a) definition of data, (b) content of data, (c) usage of data, and (d) control over access to data.

After the relationships of organizational unit to data class are established, the study team maps organization problems, organization objectives, applications planned and installed, and databases planned and installed to the data classes.

BICS AND EAS-E

The BICS methodology described above was originally implemented manually. Apart from the sheer volume of data to be analyzed, there was also a desire to compare many organizations and suborganizations. For these and other reasons, the BICS model was implemented in EAS-E.

EAS-E is an application development system based on the entity, attribute, and set (EAS) view of system modeling. Webster’s Unabridged Dictionary (second edition) defines an entity as “that which has reality and distinctness of being either in fact or for thought. . . .” In an EAS-E representation, it is usually some thing (account, check, job) of the real world to be represented in the database. It may also be a less tangible thing, like a task to be performed, which we find convenient to postulate as an entity.

The attributes of an entity can be considered to be its properties or characteristics. At any instant in time an attribute has at most one value; it may also be undefined.

In EAS-E, a set is an ordered collection of zero, one, or more entities owned by some entity. To illustrate, each account owns a set of current transactions, i.e., checks and deposits that have come in since the last monthly statement. Thus, the account for John Smith owns one such set, that for Mary Jones owns another such set, etc. In general, we say that each account owns its current transaction set. Accounts may also own a set of old transactions, outstanding loans, etc.

An entity may have any number of attributes, own any number of sets, and belong to any number of sets. An entity can both own a set of a given name and belong to a set with the same name.

These general facilities allow more specialized structures, such as trees and networks, to be expressed more or less
trivially within the general framework of entities, attributes, and sets.

EAS-E requires that the system to be implemented be modeled in terms of the entities, attributes, and sets that describe its status at a point in time. The EAS-E procedural language and nonprocedural facilities can then be used to manipulate these entities, attributes, and sets in main storage and in the database.

The BICS implementation consists of (1) programs that generate data classes from the business functions and (2) programs that capture, display, and analyze the accountability information.

A BICS STUDY

A large U.S. corporation was concerned about how well its information systems (IS) plans were tracking the corporation’s business plans, and furthermore how well IS responded to changes in the business environment. The corporation’s worldwide distribution headquarters was selected as a test. The goal was to come up with a transferable management tool that would analyze the relationship between the IS plans and the corporation’s business plans.

The distribution study differed from most other BICS studies in that the area examined was a headquarters function and not an operational area. Headquarters did not move and store products, but did the overall planning for such activities. Normally, BICS information models are related to the organizational units of the entire business. This provides the analyst with the means to analyze information flow as data passes from one organizational unit to another and to identify areas where the data are potentially out of control. In the distribution example the information models were related to the functions of distribution. At headquarters there was no people or organization involvement in the information flow, and thus there could be no identification of potential data management problems.

The internal functions identified for distribution were distribution support (this included IS and engineering), plans and control, purchasing, domestic traffic, export, import, and warehousing. The external functions identified were government, corporate finance, product service, sales, manufacturing, suppliers/vendors, and other (this included all other corporate functions outside the boundaries of distribution, such as personnel). Product service, sales, and manufacturing were, in BICS terms, the customers of distribution: They placed orders on the distribution function.

All the orders received by distribution were either for paper work to support movement of products, in which case the report or form is the product, or for the movement of the products, which is a service. Some of the orders were Request to Export, Generate Proforma, Storage Instructions, and Movement Request. For each order, the eight order management questions were asked and an information model was built. A “Yes” answer to a question generated a set of data classes from the generic BICS model stored in an EAS-E database.

The relationship between the information models and the internal and external distribution functions was now established by determining who was accountable for each aspect of each data class.

Using EAS-E, we were able to deepen the information analysis by modeling individual distribution functions. For example, we expanded warehousing into four subfunctions and built, through order analysis, information models to show information flow within the warehousing function. This was done for several other distribution functions as well.

To determine whether the BICS models could be used as a bridge between the business and IS, we took a strategic statement from the corporation’s business plans and related it to the information models. Since we had earlier built relationships between the information models and the distribution functions, we were able to determine which functions were affected by the strategic statement. If the area of the business studied had installed or planned applications, we were able to analyze the impact of the business change on existing and planned software.

The results of the study were positive. The management of the distribution function is carrying on the work, and we are now expanding BICS to enable us to generate more complete information processing requirements. This expansion is discussed in the following section.

GENERATING INFORMATION PROCESSING REQUIREMENTS

To generate requirements, the process begins, as before, by identifying the types of orders received by the organization. The eight order-handling questions are asked for each order. The answers to these questions identify the business functions. The business functions, in turn, determine the data they require and the information processes required to set up and maintain these data. These are based on the BICS data classes that have been validated in several studies.

REQGEN generates requirements on the assumption that the data are stored as entities in a central database. In other words, it generates definitions for entities, attributes, and sets that must be maintained; and for each process it specifies the operations that it performs on the EAS structures. These may be considered conceptual requirements—i.e., the data definitions can be translated into alternate-access method or database representations, and the processes can be implemented as interactive or batch programs in a conventional programming language. Alternately, the specifications can be considered EAS-E specifications: The data can be stored in an EAS-E database, and the processes can be implemented as EAS-E programs that operate directly on EAS structures.

REQGEN generates the entire data processing requirements of the organization or suborganization for which the analysis is made. If the organization has no data processing facilities, all the software can be created expeditiously from the REQGEN specifications. In most cases, however, the organization will already have some software in place. The REQGEN specifications will, therefore, be used to define changes that must be made incrementally to existing software.
The Data Definitions

As mentioned above, the data required depends on the business functions. For example, every organization must maintain entities that represent EMPLOYEES and ACCOUNTS. Organizations that select the tracking option—i.e., initiate service, change, or recall of the product after it leaves the organization—must also maintain entities that represent DELIVERED PRODUCTS.

The data definitions specify attributes of each entity type and the mode (integer, text, etc.) of each attribute. They also specify the sets that the entity type owns and the sets it belongs to. The generic definition for entity type VENDOR is shown below:

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>MODE OWNs</th>
<th>BELONGS</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDRESS</td>
<td>Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATING Class</td>
<td>Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFFERINGS</td>
<td>Product, price, qty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLANNED ORDERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTSTANDING ORDERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYMENTS OUTSTANDING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDORS</td>
<td>Owned by product</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The mode class identifies a class of attributes. Members of this class are specified elsewhere.

The organization may use these definitions as they are provided by REQGEN, but it will usually want to modify them to some extent. Let us consider an example. REQGEN recommends that the entities of type EMPLOYEE have a class of attributes called personal attributes, whose function is to store personal information about the employee. This class includes the attributes NAME, ADDRESS, HOME TELEPHONE, and EMERGENCY CONTACT. A particular organization may choose to add NUMBER OF CHILDREN to this class. Or it may choose to delete some of the attributes recommended by REQGEN.

The organization also has the prerogative of changing names of entity types, attributes, and sets. This seemingly minor facility is, in fact, of great importance. The entity, attribute, and sets generate a representation of the organization. The entities in the database represent real entities in the organization. Thus, it is beneficial if data can be referred to by familiar names—e.g., WORKER instead of EMPLOYEE or COMPENSATION instead of SALARY.

There are still other degrees of freedom in the data definition phase. The organization can choose, for reasons of convenience or security, to maintain some of its information manually. The information processes that maintain this information also become manual. A computer-based process that uses this information would need to have it entered, perhaps from a terminal.

Two other decisions must be made at this time: The organization must decide whether it will run essentially separate systems for each order type, a combined system for all order types, or some combination. Organizations that have several plants must decide whether to segregate the data geographically. If the various plants handle different kinds of orders, these two decisions become interlinked. After the organization decides how the data must be aggregated, REQGEN prepares data definition files for each separate system.

To recapitulate: On the basis of the business functions for each order, REQGEN recommends the data to be maintained. The organization modifies the data according to its business needs and policies and decides how the data should be aggregated. The final data definition files can now be produced. After this is done, the data definitions can be translated into records and fields, relations, or entities and relationships, as desired; or they may be left as entities, attributes, and sets.

The Processes

There are three basic kinds of processes: processes that set up the data structures, processes that maintain the data structures, and processes that report the current state of the database.

REQGEN provides specifications for the setup processes and the maintenance processes. It does not provide specifications for the report processes. Report processes read data from the database and present it, suitably formatted for management action. Their timing, content, and, especially, format are dependent on the individual manager and are extremely variable. A query language can be used to generate many of these reports. If a program does have to be written, report programs are often the simplest programs to write.

The few processes that set up the database are used initially, for example, to set up the various ACCOUNTS. Subsequently they are used infrequently—for example, as when a new ACCOUNT is added. Setup programs are particularly simple in structure. Typically, they create an entity, give values to a few attributes, and file it into a single set. Because of their simple nature, EAS-E programs to implement these processes could be generated by a future version of REQGEN.

The majority of the process specifications generated by REQGEN are for maintenance processes. For each process REQGEN specifies generic operations to be performed on the EAS structures—i.e., the entities that must be created or destroyed or filed into or removed from sets, and the attribute values that must be updated. In most cases it cannot specify the algorithm to carry out the updating, since this will depend on the nature of the organization and its style of management.

For example, every organization that manufactures the product must have a process to produce a production plan. This process gives values to attributes that represent the production of each product by month, quarter, or other appropriate period. The production planning process must also give values to these attributes for every component of each product. Some of these components may, however, be purchased instead of being manufactured. If the component is purchased, the process must give values to attributes that specify the vendor, the quantity, etc. For each process REQGEN provides a text description similar to the one above. It also provides a list of the entity type it creates, the entity types it destroys, the attributes it reads and writes, and the sets it files and removes from.
REQGEN specifies that the production planning process is necessary if manufacturing is the type of work being done. It also specifies the entities, attributes, and sets that it works with. It does not insist that the process be carried out on the computer, and it does not specify the algorithm that must be used. A job shop may use a bill of materials explosion, whereas an oil refinery may use a linear programming approach.

Each process is triggered by an event—an external one, such as the receipt of an order or a payment, or an internal one, such as a change in product formulation or the generation of a facilities plan for the next year. Some of these events are periodic—quarterly preparation of financial statements, annual physical inventory of facilities, etc.

When an event occurs, the programs that implement the appropriate processes must be invoked. This is generally controlled manually, and manual procedures must be put in place to do so. REQGEN specifies the events that trigger each process and the organizational unit responsible for it, but it does not address the installation management procedures.

SUMMARY

Starting from an identification of the types of orders received by the organization, REQGEN generates data definitions in terms of entity, attribute, and set structures. It also generates specifications for the processes required to maintain the data and lists the events that trigger each process.

FURTHER WORK

The next step is to validate and tune REQGEN in real organizations. In a more theoretical direction, we would like to understand the causes underlying the variation in process algorithms and attempt to specify them insofar as possible. We know that one cause of variation is technology, and we feel that we can specify some of the algorithms for some of the technologies. If the algorithms can be specified, then it is not very difficult to generate the actual programs. There are precedents in this direction. Markowitz describes a job shop simulation generator that captures most of the complexity (excluding accounting) of activity scheduling in a job shop. The IBM Application Customizer also went a long way in generating commercial programs for specific industries. Our ultimate goal is to prepare a complete package of management software based on a description of the goals and policies of the organization. REQGEN is a first step in this direction.

REFERENCES