Reverse Engineering: Its Use on a Legacy System

by William Aldrich

Recent years have been tumultuous for M. As the broader software community has accepted and greatly improved upon many techniques and features that have been commonplace for M, M users have faced increasing criticism and competition that question its relevance and future. One can summarize many of those complaints into three broad, but interrelated, weaknesses.

First, the documentation for older systems exists on paper or in M globals, if it exists at all. The quality can be poor. An organization has to commit significant resources to time-consuming and often frustrating research to rectify the problem. Managers and programmers who wish to make their organizational mark in an exciting new project are not likely to join or even to support such an effort. Second is maintaining existing software. Systems with long life spans create many issues. Applications have users who are very comfortable with the existing system. No one wants to risk changes that can offend users. Programmers and analysts work with older M systems that typically do not incorporate a data dictionary. Over time multiple locations contain the same data for different purposes because the technical staff lacks the time and means to create and to update a data dictionary. Managers have increasing pressures to keep changes within budget. Third, older M systems have been through many changes in management and staff. Different or non-existent styles of management and coding have created chaos in programs and in documentation, making it difficult to make enhancements.

Given these weaknesses, what can an MIS group do to prepare itself for the future? One example is a large health maintenance organization (HMO) in the Northeast with a legacy CoStar (Computer Stored Ambulatory Record) system. A technical manager initially thought he could write a few "one-shot" programs that used the results from two utility tools to resolve the above problems and reduce a routine set that had roughly four thousand routines. Those tools are the M routine %RSE that lists the occurrence of selected strings of characters within a given routine set and the M routine %INDEX that shows within a given routine set all syntax errors, occurrences of globals and local variables, and called routines.

Fortunately another alternative exists. The technical manager selected RE/data from George James Associates (England), which Greystone Technology markets and supports in the United States. RE/data provides a variety of information. It also has the capability to link to DBMSs (M/SQL and other SQL databases) and to CASE tools (System Architect). Within four months these tools provided distinct results that reduced the routine set by nearly 50 percent and improved the staff capability to support the CoStar product.

Using these tools is not necessarily a quick panacea. CoStar is a product that contains many customized features. RE/data is not well documented. Users must work with each other to discover how to use keyboard keys, and to understand the flow of information from the data repository to the data dictionary. Once the users overcome the steep learning curve, RE/data begins to produce significant benefits. It automatically populates a data dictionary with global references along with metrics such as the minimum and maximum length of an element and if possible the local variable commonly associated with the element. The user can add descriptive text to each element.

RE/data treats each global node as a structure. As it builds the data dictionary, it collapses the related nodes. A reporting function displays information on the existing system by global, node, and element. One sees backward and forward references to each element on a node, and the foreign key referenced in each node. It is easy to find the data elements under a node and then the nodes belonging to a global.

After creating and documenting the data dictionary, one can move it to other database tools. When moving to M/SQL, RE/data automatically creates a table from each collapsed node. One can also write a custom exporter that creates a CSV (comma separated file) to other database products, such as KB_SQL or the desktop tool of one's choosing. It is possible to write a custom exporter to a CASE tool called System Architect (I have done so), which helps to build a model of the database. One can use System Architect to automate various aspects of the conversion to a new medical record application, also written in M.

Using reengineering tools such as RE/data is not easy, but the financial and technical paybacks are considerable. **M**

Bill Aldrich works with M in billing, payroll, ADT, and medical records. Outside of M, he contributes articles on mountain biking and railroads. His address is WALDRICH@WORLD.STD.COM.