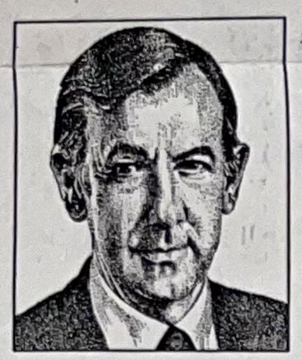
APPLIED INTELLIGENCE

Making the CASE for True Reverse-Engineering Tools



JAMES MARTIN

To help maintain and enhance the billions of lines of existing application code, programmers need powerful reverse-engineering tools that can both derive process models and data models from existing code, and help rebuild these applications within a more modern distributed-processing environment.

Reverse-engineering is a software technology that has as its goal the migration of old systems into a cleanly engineered form that can be enhanced easily. The reality is that true reverse-engineering tools that convert existing data definitions and procedural code into high-level specifications are not yet available.

For instance, computer-aided software engineering (CASE) tools are oriented toward the specification and implementation of new applications, and thus fall short of providing much assistance in the support and improvement of existing application code.

There's a lot of confusion in the field about what is meant by reverse-engineering. To clear up the confusion, consider the three categories of tools that support various components of reverse-engineering:

- **Restructuring Engines.** These tools automatically convert unstructured source code (such as COBOL) into structured code. The goal is to reduce maintenance costs, improve quality, extend the life of old systems and enable program enhancements to be made in much less time. The process automatically creates a structured version of the original source code without changing its functionality.
- **Re-engineering Tools.** These tools analyze the source code to identify its structure and detect data redundancies, non-standard names and unused code. Documentation is produced that lists cross-reference tables, structure charts and relationships between files, records and fields. An analyst can interact with the tool to resolve inconsistencies and restructure process or data definitions. The standardized definitions can then be stored in the repository of a CASE tool for further analysis.
- **Reverse-Engineering.** The term "reverse-engineering" should be reserved for tools that are capable of synthesizing unstructured data definitions and processing definitions up to the level of pure specifications, independent of environment. These specifications can then be used in a forward-engineering process to reimplement the application in the same or a different environment, using standard CASE techniques. Extraction of specifications from existing data definitions and process code is an extraordinarily difficult task, which has been solved, to date, only on the data side.

Most CASE tools that claim reverse-engineering capabilities actually support re-engineering functions. They don't synthesize low-level data and process definitions up to the level of environment-independent specifications.

The figure illustrates the functions performed by a true reverse-engineering tool. Reverse-engineering of data functions are shown on the left of the figure; reverse-engineering of process functions are shown on the right.

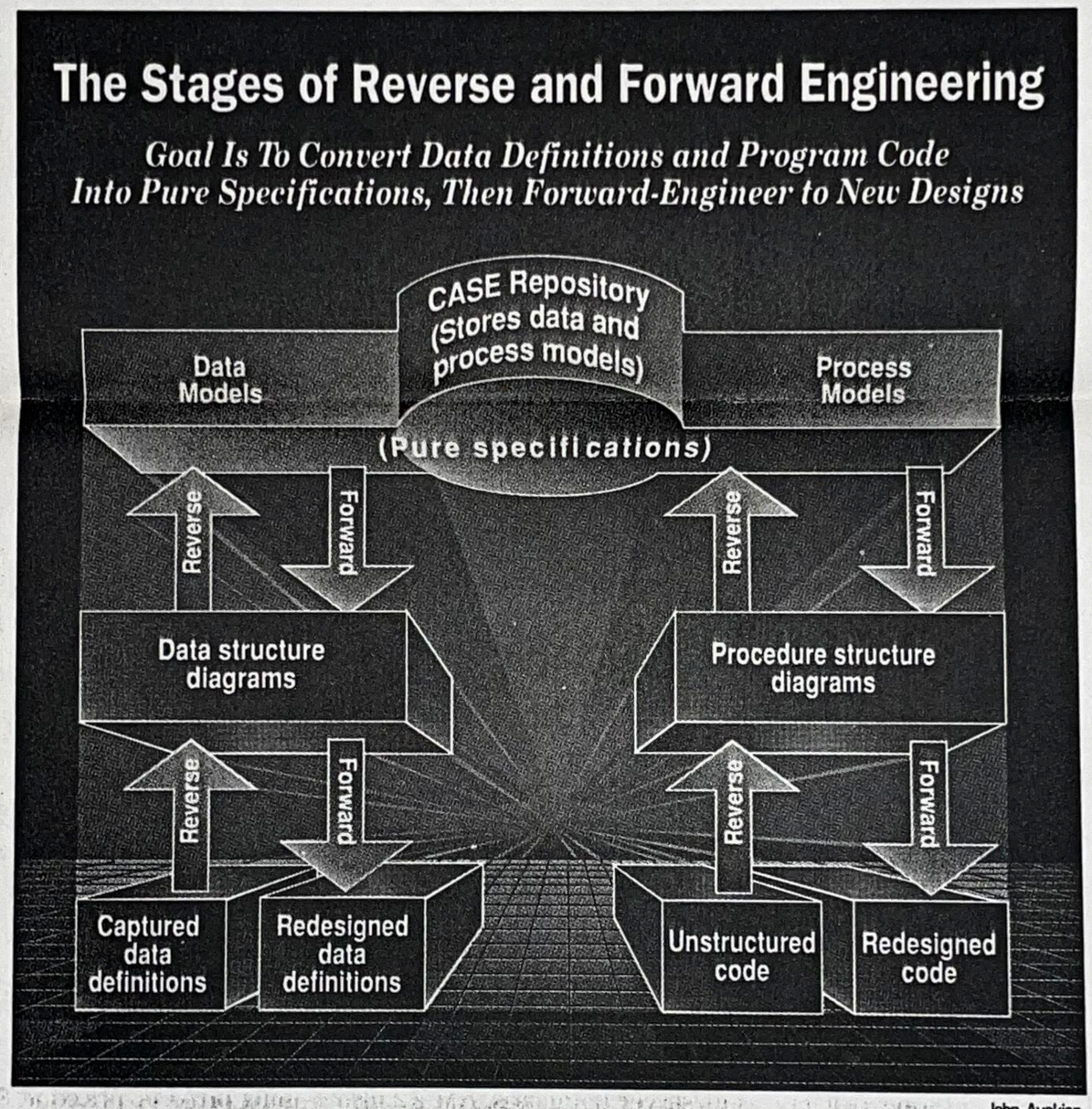
As shown, conversion of data definitions into specifications is done at three levels: capture of physical data definitions in the program; development of a schema description contained in a data-structure diagram; and conversion of the data-structure diagram into a high-level data model (or conceptual schema).

Low-level data definitions in the program are first captured from existing files and analyzed using a form of re-engineering. A programmer can interact with the tool to resolve inconsistencies and produce more uniform data definitions and data-element formats. However, if data definitions from many applications are loaded into the tool, the mess revealed may be so great that the programmer is discouraged from cleaning it up. Thus, it's generally desirable to tackle a small area at a time.

The result of this analysis is the conversion of data definitions into an intermediate form called a data-structure diagram. The data in the old program is likely to be unnormalized. The data-structure diagrams need to be normalized and the normalized structure made to conform to a data model.

The data model is a pure specification, typically expressed in the form of an entity-relationship diagram.

At this point, the data definitions from the program have been converted into a data model, at the specification level. From that data model, new data structures can be created for the same database-management environment or a different environment.



John Avakian

The reality is that true reverse-engineering tools that convert existing data definitions and procedural code into high-level specifications are not yet available.

The reverse-engineering of the data side has been implemented by Bachman Information Systems Inc. in its Bachman/Re-Engineering Product Set. This product set is capable of capturing data definitions in IMS, IDMS, DB2 and VSAM systems and then migrating these designs from one database-management system to another.

The product set can also be used to create data models for new applications, which can then be migrated via direct links to other CASE tools.

The Bachman tool set is implemented in the form of a large expert system that provides assistance by indicating to the programmer or analyst what information is missing to create a data-structure diagram or data model. The programmer fills in the missing pieces of information while interacting with the system.

Reverse-engineering the process side, which requires converting unstructured program code into high-level design specifications that can be input into the repository of a CASE tool, is proving very difficult. The application may be written with spaghetti code; it needs to be converted to structured programming.

Several tools exist for automatically restructuring COBOL programs. The product Recoder from Language Technology Inc., for example, converts COBOL programs that are messy, unstructured and badly coded into fully structured COBOL.

The resulting program is functionally identical to the original. The restructured code can be tested, perhaps with the original test data, to ensure that it's functionally equivalent to the original.

The restructuring process produces structured source code but not process-model specifications. To support reverse-engineering on the process side, it's necessary to convert the structured code into high-level specifications that extract the business functions performed by the code. These specifications form the basis for enhancing and maintaining the application.

Action Diagrams

After the program has been converted to a fully structured form, it should be synthesized into a set of action diagrams for a CASE tool. Action diagrams, which define the process model for the application, can be converted into other diagram types (such as decomposition diagrams, structure charts or data-flow diagrams) and can be linked to data models, screen designs and so on.

Although the structuring of source code can be done automatically, the extraction of design specifications from the structured code requires the assistance of an analyst. The analyst interacts with the system and fills in missing data, such as logical groupings of business processes, refinement of data flow and consistent naming conventions.

No CASE tool currently supports reverse-engineering of unstructured source code into process models at the specification level. The introduction of such a tool would be an invaluable addition to the CASE market.

Next week, I will discuss tools that support re-engineering of existing source code. ■

The concepts embodied in reverse-engineering are described in the CASE volume in The James Martin Report Series. For more information on this volume, call (800) 242-1240. For information on seminars, contact (in the United States and Canada) Technology Transfer Institute, 741 10th St., Santa Monica, Calif. 90402 (213) 394-8305. In Europe, contact Savant, 2 New St., Carnforth, Lancs., LA5 9BX United Kingdom (0524) 734 505.