Payoff

Reengineering is a vital process for reclaiming past information systems investments. Although current CASE tools provide only partial solutions, they can nonetheless help with systems reengineering efforts. This article includes a listing of tools available to help IS reengineer existing applications.

Problems Addressed

One purpose of reengineering is to provide IS departments and professionals with a way to extend the useful life of their huge investment in existing systems. Reengineering helps organizations examine and modify existing information systems to meet the changing needs of the organization. Traditional systems development progresses at a very slow pace. Reengineering helps organizations to rethink systems development activities in order to support a more rapid implementation of higher quality systems.

This article discusses the use of Computer-Aided Software Engineering tools for reengineering existing applications and analyzes the capabilities of tools currently available. Implementation considerations and guidelines are also discussed.

Reverse, Forward, and Reengineering

The terms reverse engineering and reengineering are often used interchangeably even though they do not mean the same thing. Likewise, the term forward engineering is also misused. Before discussing the use of Computer-Aided Software Engineering tools for reengineering existing applications, it is important to clarify these terms.

Reverse Engineering

Reverse engineering is the process of analyzing a system and representing its components and interrelationships in a more abstract form, independent of the physical implementation. It involves recovering design specifications or requirements from existing source code or file definitions. Typically, some sort of graphical representation is the result. The better tools available for reverse engineering retain this information in a data dictionary or knowledge base.

Reengineering

Reengineering is the examination and alteration of a system to reconstitute it in a new form. Once a system has been reverse engineered into an abstract form, reengineering is the modification of this abstraction in order to meet new needs, make corrections, or take advantage of opportunities.

Forward Engineering
This is the process of going from high-level abstractions (implementation independent) to the physical implementation of a system. It is the construction and implementation phase of a typical development methodology. Exhibit 1 illustrates the difference between a reverse engineer process, reengineering, and forward engineering.

**Difference Between Reverse Engineering, Reengineering, and Forward Engineering**

**New Development Versus Maintenance**

There are Computer-Aided Software Engineering tools to support structured analysis and design methodologies, proper data analysis and modeling techniques, and enterprise modeling. However, by focusing exclusively on these development efforts, organizations sometimes ignore maintenance and support activities, which make up the largest resource expenditure in the IS department. IS organizations spend anywhere from 60% to 90% of their time, effort, and money maintaining, enhancing, or adapting existing computer systems. Roughly 80 billion lines of Common Business Oriented Language code amounts to a huge financial investment to protect. The federal government alone spends more than $5 billion a year on software maintenance.

Technological research and methodology developments are, by their nature, oriented toward new systems, however. Generally, it is new systems development that uses new technology that receives the most management attention. The IS culture does not tend to reward achievement in maintenance.

In many cases, though, more effort goes into changing programs than into writing them in the first place. As a result, it can be more cost-effective to redesign and rewrite existing applications rather than to try to reengineer them. To ignore the investment in past information systems and start from scratch is, in effect, like reinventing the wheel. At an estimated development cost of from $10 to $25 per line of COBOL source code, a complete system rewrite can seldom be justified.

CASE tools for reengineering are designed to assist analysts in understanding existing systems and to make it easier and less costly for them to make changes. Most of the Computer-Aided Software Engineering tools mentioned in the Appendix bring existing systems from their physical implementation level into structured analysis and design constructs. From there, they can be logically enhanced or corrected, then forward engineered as though they were the product of development rather than of maintenance.

The benefit of this approach, according to industry studies, is that a reengineered project can reduce maintenance costs by 30% to 70%. Some organizations that have implemented Computer-Aided Software Engineering reengineering tools report remarkable productivity boosts (as high as 300%), particularly in reverse engineering of files and data bases.

**Choosing the Right Reengineering Tool**

Many Computer-Aided Software Engineering vendors have added capture and reengineering tools to their existing top-down products. CASE reengineering tools are available for a variety of applications. Although the list in the Appendix is not

---

comprehensive, it includes the most popular and more recognized tools for reverse engineering and reengineering. The list is, however, only a subset of all the products that are (or claim to be) reengineering tools.

### Identifying Top Maintenance Activities

One way to begin the search for the right CASE reengineering tools is to examine where the IS organization spends most of its time in maintenance. Adaptive maintenance, which is done to respond to changing user requirements, consumes from 49% to 70% percent of IS resources in many cases. Of this, about 40% of a programmer's time is split between understanding the change and tracing the program's logic. Another 28% is spent in testing and debugging the change.

These statistics suggest that the biggest return on reengineering investment would be in code analysis and testing tools. This is further supported by a sampling of programmers and managers who enhance and maintain existing mainframe systems. Out of 17 different CASE tool categories they rated, the number-one and number-two rankings, in desirability, were program testing and program analyzing tools, respectively. The use of program testing and analyzing tools can help to stabilize the current environment and provide increased productivity in programmer maintenance, but this is just the first step.

### Analyzing Tool Features

In addition to windowing capabilities, word processing functions, multiple user support, and enhanced project management, all of which are desirable features in Computer-Aided Software Engineering tools, there are several other important considerations that should influence tool selection.

#### Underlying Structure.

Data base management system (DBMS) or dictionary-based software system tools provide for greater reengineering and forward engineering capabilities. These systems are also better able to accept future capability enhancements.

#### Cooperation and Direction of the Vendor.

The vendor should be willing to provide a list of installations using its software and give potential buyers permission to contact them. If a vendor refuses, its product cannot do what it claims and it is not worth pursuing.

The future direction of the tool and the vendor must also be examined before excluding a tool from consideration. For example, are desirable features expected to be available, and is the vendor making significant investments (30% or more of revenue) in research and development?

#### Compatibility and Integration.

Among the issues here are whether the vendor has an open architecture philosophy and effective interfaces with other CASE tools already purchased or under evaluation. Business partnerships, such as those between Bachman Information Systems,

---

KnowledgeWare, InterSolv, and IBM Corp., usually result in toolsets that follow the same (or similar) communication protocols and therefore are usually more integrated.

**Methodology.**

The tool should be able to be used relative to the organization’s internal or existing methodology (if one exists). If the organization does not follow a methodology, then it should consider whether the tool has one built in. The more flexibility a tool has in this regard, the more control IS has. A tool that has its own methodology built in may be less flexible, but it will tend to do a better job of leading developers through the process. This may be of significant value in an organization with little experience, a weak methodology, or a high turnover of IS personnel.

**Interactivity.**

Interactive tools should be considered ahead of black box approaches to complete reengineering. Until artificial intelligence in CASE tools becomes more sophisticated, programmers and analysts must add something to the process of reengineering. Common Business Oriented Language restructuring tools, in particular, are a black box reengineering approach. Vendors claim these tools can produce a functionally equivalent structured program from even the most tangled of source code, but it is important to distinguish between structured code (syntax) and structured program (logic). Restructuring tools can generate syntactic structure, but they are still incapable of logical or functional structure. Until Computer-Aided Software Engineering tools advance to where they can consistently make logical inferences from illogical process flows, this task is best left to humans.

**Best-in-Breed Approach to Implementation**

Better practices in systems analysis and design result in systems that are highly cohesive and loosely coupled. It is wise to approach reengineering in a similar manner. A collection of tools, each oriented toward providing just one of the many kinds of information required for complete reverse engineering, is probably the most effective strategy to follow.

This best-of-breed approach does not necessarily mean that the tools should come from different vendors. A single vendor may have several specialized tools to offer and the tools are likely to be well integrated. If the selected tools are integrated, they are likely to provide a more satisfying solution then a single tool that is, in effect, a jack of all trades and master of none.

**Recommended Course of Action**

As systems maintenance, enhancement, and adaptation pressures continue to mount, more organizations will need to adopt some type of reengineering plan. Although reengineering can be practiced with current technology, the achievement of many of its benefits may require additional automation. Systems professionals need not and should not wait for full tool support. Despite the partial solution to automation that Computer-Aided Software Engineering tools provide, it is better to reengineer with CASE tools than without them.

Information technology provides many options to reorganize work, but apart from the technology, managers must also have a well-developed strategy for its implementation. Some large corporations are facing major problems and failures with reengineering
because of improper implementation strategies. To succeed and achieve the improvement in performance, productivity, and profitability that is the goal of reengineering, the following implementation activities should be followed:

- **Defining benchmark objectives.** This means determining how to make the existing system more responsive and available to the changing needs of the business, then devising a strategy to implement the vision and ensure proper handoffs.

- **Securing executive commitment and support.** Top executives must set directions and establish commitment. Without the top-down commitment of executives, it is difficult to redirect the momentum of the organization.

- **Maintaining project scope definition.** Making significant systems changes late in the game, after requirements are supposedly frozen and work has begun (an occurrence sometimes known as scope creep), is as detrimental to redevelopment as it is to new development.

- **Developing a comprehensive inventory of JCL, programs, and files.** This inventory should be used to set the project scope boundaries.

- **Controlling management's expectations.** Expectations should match reality. Even with CASE tools, reengineering is going to be a knotty and complicated process.

- **Training IS workers.** The benefits of reengineering have to be conveyed to all concerned IS workers so that they can become the companywide coordinators (enablers) of the process.

- **Emphasizing a process orientation.** Businesses should think in terms of cross-functional processes rather than traditional stovepipe functions.

- **Being prepared for surprises.** The number and severity of these surprises are often directly related to the age of the system being reengineered.

**Bibliography**


Author Biographies

David B. Borchardt
David B. Borchardt is manager of systems projects at International Paper in Memphis, where he is responsible for corporate financial and accounting applications. He holds a degree in computer science from the University of Wisconsin at LaCrosse.

Narender K. Ramarapu
Narender “Nari” K. Ramarapu is an assistant professor of management information systems and production operations management at the University of Tennessee, Martin TN. He is a member of the Decision Sciences Institute, The Institute of Management Science, and the American Production Inventory Control Society.

Mark N. Frolick
Mark N. Frolick is an assistant professor of management information systems and the associate director of the Federal Express Center for Cycle Time Research at Memphis State University. His research interests include executive information systems, systems analysis and design, outsourcing, and telecommuting.