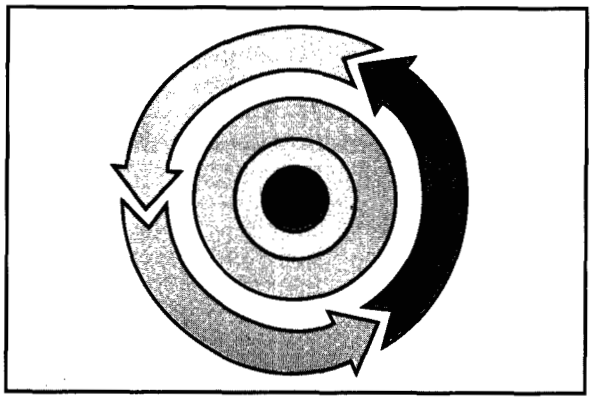
J. R. Tirso, "Establishing a Software Reuse Support Structure," *Proceedings of the IEEE International Conference on Communications*, Denver, CO (June 1991), pp. 47.2.1–47.2.5.
W. Tracz, "Software Reuse: Motivators and Inhibitors," *Proceedings of Computer Society International Conference*, (COMPCON), Spring '87; IEEE Cat. No. 87CH2409-1 (February 1987), pp. 358–363.

J. R. Tirso
IBM Personal Software Products Division
Boca Raton
Florida

H. Gregorius
IBM Large Scale Computing Division
Poughkeepsie
New York



Information reuse parallels software reuse

Organizations that place a high value on their information can best leverage and maintain that information if they apply the same infrastructure and techniques used for reusable software. The software life cycle produces several types of reusable information such as customer information, product development information, and process information. Just as software reuse benefits from structured programming practices, information reuse benefits from the use of common tools, centrally coordinated standards and terminology, and development practices consistent with good writing and design.

Reusable information. Information reuse is the reuse of nonexecutable entities. We distinguish information reuse from information management by the end use of the retrieved information. In information reuse, the user incorporates the retrieved information into a work product, whereas in information management, the user retrieves information to read or reference it. Computerized library card catalogs and information retrieval tools are examples of information management systems.[1]

The nonexecutable entities in information reuse can be grouped into four categories because of differing characteristics and requirements: customer information, process information, product development information, and miscellaneous information. Customer information describes software to the customer. Process information focuses on the process, such as ISO 9000 documentation, process diagrams, schedule documentation, and quality projections. Product development information includes business cases, requirements, designs, test cases, and plans. Miscellaneous information includes reusable forms and product-independent graphics.

Further complicating the issue of information reuse are the numerous media in which businesses can deliver the above categories of information. Four such forms are hardcopy, softcopy, integrated on-line, and hypermedia. Hardcopy consists of text and graphics printed on paper; softcopy is this same material when displayed on a video display terminal; on-line information consists of all text and graphics stored with the code for display and use in an integrated, interactive manner; and hypermedia is text, graphics, animation, audio, video, image or executable code stored in various places and logically linked together.

To determine whether a piece of information should be supplied as a reusable part, three questions may be asked:

- Is there a known near-term need for other uses of this information or document?

- Is the document structured so that it can be tailored with relative ease?
- Is there a commitment to maintain the document once it is available for reuse?

Before an organization supplies a part, it must decide the most effective structure for the information. In contrast to code reuse, in which the use of unmodified parts is stressed, information reuse often involves parts designed for modification by the user. Examples of information intended for modification are fill-in-the-blanks-type reports, document outlines, and shells or templates for displays or forms. We still encourage reuse of unmodified information parts when the situation permits, such as in the case of diagrams, "clip art," and hypermedia nodes.

Two effective ways to structure reusable information are (1) as an *example* (complete information) or (2) as a *template* (framework for the information). These terms are used to distinguish between the two levels of information completeness in reuse libraries. Example information may be a complete document, graphic, or screen that provides the user with an integral picture of the information entity. The user can retrieve and use the example as it is or modify it to meet a specific need. The user may also retrieve and use the example to get a better understanding of how to fill in a template. Templates are documents, graphics, screens, and other items that have a framework for the generic sections of the item and instructions on how to complete the reusable item. From these templates and examples the user has the potential to produce customized quality items in less time. The systematic use of templates and examples can result in dramatic savings.[2]

Infrastructure for information reuse. As for software reuse, the infrastructures used by IBM[3] and Hewlett-Packard Co.[4] provide cross-project leverage. Just as sites may have reuse champions to foster reuse at a laboratory, and projects may assign reuse leaders to encourage reuse on a product, information developers should have an *information reuse champion* to encourage reuse of all information items. It is important for the information developers to have adequate representation within the central coordinating organization so that information-specific issues and nuances can be addressed by the overall reuse policies.

There are certain aspects of the reuse infrastructure that a central organization typically handles. These include establishing standards to facilitate reuse of parts across organizations. The standards must include requirements for specific quality levels within the reuse libraries. The standards should also include content standards, or references to content standards (e.g., tagging and style standards used by publishers to define the layout of documents), and interface standards (e.g., definition of how to make reusable information parts interconnect or flow together).

Experience shows that the classification terminology also needs to be managed centrally. Without central coordination, each discipline or business area uses separate and distinct terminology, which can have two potential impacts on users. First, the users may be overwhelmed by the number of terms used in a classification and, second, the users may have difficulty discriminating among some of the terms because several terms may represent the same concept (depending on the domain) or the same term may represent different domain-specific concepts.

All of these situations reduce the likelihood of a user successfully retrieving applicable parts.[5]

Measuring the value. We can make the case that information reuse saves money and time, but there are still obstacles to overcome. As with software reuse, we must have the management support necessary to begin measuring the payback of information reuse. For example, graphics and hypermedia are information areas with a high potential reuse return on investment because of the high cost of creating new information. The first step is to get management to recognize the benefits to the business through the use of modified software reuse metrics or other information value techniques.[6] Organizations involved in process-related initiatives, such as total quality management or ISO 9000 registration, will already recognize the value of process information. Managers whose product sales require user-friendly documentation should also quickly recognize the value of information.

"More isn't always better" summarizes the dilemma for reuse metrics. Organizations typically reward productivity of code and information based on how much an individual produces—the more lines of code, function points, or text words

per unit of time, the better the results in terms of productivity numbers. Information reuse metrics in particular should reflect the value added to the product. The goal is to devise metrics that reward the developer for the closeness that is achieved to the goals of the product release. For an operating systems programmer, less may be highly preferred. For an information developer, the amount of information is less important than whether the end user can easily understand the concepts or tasks being discussed.

There are no perfect solutions to the reuse metrics problem. Although industry recognizes a number of issues related to measuring reuse,[7] IBM currently has accepted software reuse metrics.[8] We also have an initial set of measurement standards for information reuse and will continue to evolve these standards as experience dictates. Based on this prototype framework, most current projects show information reuse percentages (how much information is reused) in the single-digit range, which we find healthy and predictable at this early stage in our information reuse program.

Another consideration related to business and measurements comes from the intangible benefits of reuse. In software, these intangible benefits consist of delivering more function per release or improving performance and quality with reused parts. In information reuse, the intangibles include quality improvements such as additional indexing, enhancements to technical descriptions, and increased time spent planning for the next release. In contrast to software reuse, information reuse does not have a large experience base from which to judge the effects of these results, so intangible benefits must be projected from a small sampling and from general experiences.

Tool considerations. For information reuse to succeed, we must have adequate tools available and those tools must be integrated into our information development environments. We must also consider other tool issues intrinsic to information development, such as the ability to manage graphics and multimedia files.

Due to the relatively high cost of producing graphics compared to the cost of producing text, graphics is an area rich with reuse opportunities. To facilitate the search and retrieval of graphics (illustration graphics and images) within the development environment for information, there should be, at a minimum, the capability to browse and cut-and-paste graphics. This implies a requirement for workstations with connections to a reuse repository.

In most organizations, information developers use a variety of products to create graphics and images with a variety of formats. Therefore, the information reuse champion should:

- Convince all information reuse participants to standardize on one or two products (and formats) and provide these products to everyone currently using other products
- Provide support for all commonly-used graphics and image products and provide translation utilities to allow conversion between the formats

Selection of either of these options may lead to a temporary decrease in productivity while training on the new products, or training on conversion, is accomplished. However, the organization should experience long-term productivity gains through the sharing of quality graphics.

Use of local area network graphics tools such as CorelDRAW** or Teamwork** can also help the casual user. Benefits are gained by reducing the time and frustration of the users because they do not have to install multiple products and maintain a consistent level of product capabilities. This makes the user's job of browsing and editing the reusable graphics much easier.

Another tool consideration is the capability to manage hypermedia parts. This broader set of parts includes all of those already discussed for graphics and images, plus those related to the use of sound and video. In addition to the difficulties associated with handling a variety of file types (and the associated requirements for storage media, browsers and editors, and education), there are the additional issues of classifying, managing, maintaining and retrieving the multitude of small parts. At this time we find the most important retrieval issues relate to performance and usability of the tools. However, we believe that in the long term, maintenance issues will be the most difficult and costly to resolve.

How to build reusable information. The creator of reusable information can achieve reusability

through understanding the desirable characteristics of reusable software and then applying those characteristics to information. These characteristics of software reusability are similar to those promoted by software engineering practices.[9] Reuse requires a focus on the basic problem of good software design and development. Similarly, reuse of information is successful only when information developers follow good writing practices. Some desirable characteristics follow.[10]

Abstraction. The component extracts only essential properties from the problem space to model in software and omits nonessential details. The component abstracts both data and algorithms.

Ease of understanding. The component is thoroughly documented, including self-documenting code and plenty of in-line comments.

Functional completeness. The component has all the required operations for the current requirement and any reasonable future requirements.

Uniformity. The component uses consistent notation, control structures, and calls, and logically relates objects the same at any level.

Modularity. The component has good structure to organize data and algorithms. Components also exhibit the desirable properties of loose coupling (few interconnections) and strong cohesion (the component does a specific and well-defined function).

Reliability. The component consistently performs the advertised function without error. The component has been repeatedly tested across various hardware and operating systems.

Information hiding. The component suppresses implementation to focus on abstraction and make levels of abstraction independent of each other. It hides implementation details from the user. The component clearly defines the interfaces to allowable operations and data.

Good error and exception handling. The component isolates, documents, and handles errors consistently. The component provides a variety of options for error response.

Portability. The component does not depend on unique hardware or operating system services.

Interpretation of some of these characteristics is needed to understand how they apply to information. For example, error handling and portability apply most directly to information written for on-line or hypermedia use. It is as important to thoroughly test on-line or hypermedia information on all platforms of potential use as it is for code. However, information for only hardcopy use has little reliance on these two characteristics to achieve reusability. Error handling for on-line and hypermedia information has the same problems as code, that is, portability across platforms and interoperability between parts. Our parts designed for cross-platform reuse have so far only achieved a partial solution to error handling.

Abstraction, uniformity, and information hiding are familiar concepts to information developers trained in hypermedia development,[11] although the terminology used here is software-based. Ease of understanding is always one of the most difficult objectives to achieve. For reuse, ease of understanding has the added burden that it must be accomplished within the framework defined by modularity.

Creating components for reuse libraries. To encourage a supply of reusable parts (components) we should give consideration to the following:

- Make supplying components easy. Grouping parts into many small components with similar characteristics must be balanced by the costs of developing and supplying the supporting information for each component. Build templates for the parts containing supporting information and develop other cost control methods for packaging components.
- Store related parts together. Sometimes we organize groups of related reusable parts into components. For instance, the design, code, test cases, and user documentation for a particular software function may be stored together in the reuse library. All the information needed to use a part should be placed together.
- Consider packaging alternatives. Large documents with many files are best kept in separate components. This will reduce the confusion when trying to locate files embedded from a particular document or component. If the document consists of a small number of files and is related by subject and owner, it may be grouped with other smaller documents into a component. An example of this is grouping hypertext link definition files for all the products at a site into one

component. Consider where files will be located and the impact of grouping documents into a component before beginning packaging.

- Make it easy to find the right part. The classification of the components will not facilitate retrievability unless the parts in the component have many similar characteristics. Put the information into the component that will meet a predicted use, not several uses. Consider the terminology of the domains where the part applies and choose the terms that most accurately and naturally describe the part. If the user cannot find the part, the user cannot use it.
- Make it easy to get the part. Consider which groups, at your site and other sites, will want to access and retrieve this reusable part. Consider how this part fits with other parts and where intended users may look for it. Provide options for users on different computer systems and development environments to easily retrieve the part.
- Assign maintenance responsibility. The definition of a component states that all information in it is the responsibility of a single owner. The owner may be a department or a delegated person, such as a librarian. However, developers should clearly understand that they have maintenance responsibilities for their component. Delegation of maintenance cost is the largest cost benefit of reuse.[8]

Classifying for retrieval. Retrievability, necessary for reusability, is aided by accurate classification of components using common terms. We recommend a dual retrieval process consisting of classifiers search and then a free-text search on the source files of the candidate parts. Recent empirical data also support this two-pass approach.[12]

Classifiers aid retrievability by describing the characteristics of a component or an entire library through the use of common terminology. Examples of classifiers are the national language of the information and the name of the author or developer. Classifiers may have a bounded set of values (terms) from which to chose,[13] such as the national languages supported, or have an unbounded set of values, such as the set of possible authors' names.

During the last two years, an increasing number of organizations and companies have experimented with classification schemes as a method for retrieving software parts.[13–15] Some have concluded that the maintenance costs and usability factors outweigh the benefits.[16] Others have tried alternative approaches to information classification[17] or information retrieval.[18] The experiences at IBM indicate that the environment (distributed versus centralized), the size of the development group, and the breadth of the domains that require searching for a particular reusable part determine the classification and retrieval requirements. This is especially true when we consider the diversity of information types and the many forms information may take.[19]

Summary. Reusing information products can extend the effects and benefits of the traditional corporate code reuse program. Just as organizations leverage their software modules and components in new applications, pieces of information in such forms as text, graphics, and hypermedia can find homes in subsequent products. However, although software reuse has managed some successes, information reuse remains in its infancy. The need to establish a supporting organizational framework, tools, and standards continues to pose technical obstacles to information reuse. This forum entry outlines how the information reuse program at IBM built on our software reuse experiences and begins to address these obstacles. Although relatively young, information reuse is a key part of our development process and reuse program.

Acknowledgments. The author would like to thank Ann Arader, Joseph Caruso, Jeffrey Poulin, and the many reviewers for their comments, insight, and help in the preparation of this forum entry.

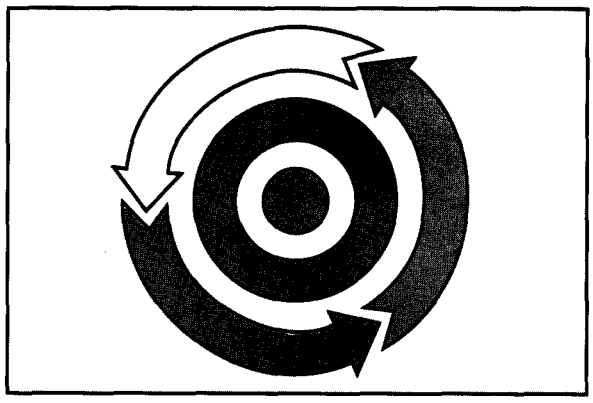
**Trademark or registered trademark of Corel Corp. or Cadre Technologies.

Cited references and note

1. G. Story, et. al., "The RightPages Image-Based Electronic Library for Alerting and Browsing," *IEEE Computer* **25**, No. 9, 17–26 (September 1992).
2. M. M. Sherry, "Methodology for Software Documentation Reuse," *Proceedings of the Human Factors Society 36th Annual Meeting—1992*, Human Factors Society, Santa Monica, CA 90406 (1992), pp. 198–201.
3. See the first entry in this forum, "Management of Reuse at IBM," by J. R. Tirso and H. Gregorius.
4. M. L. Griss, "Software Reuse: From Library to Factory," *IBM Systems Journal* **32**, No. 4, 548–566 (1993, this issue).
5. J. S. Poulin and K. P. Yglesias, "Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL)," submitted to Computer Software and Applications Conference (COMPSAC '93), Chicago, IL (November 3–5, 1993).

6. R. Glazer, "Measuring the Value of Information: The Information-Intensive Organization," *IBM Systems Journal* **32**, No. 1, 99–110 (1993).
7. J. S. Poulin, "Issues in the Development and Application of Reuse Metrics in a Corporate Environment," *Fifth International Conference on Software Engineering and Knowledge Engineering*, IEEE, San Francisco, CA (June 16–18, 1993), 258–262.
8. J. S. Poulin, D. Hancock, and J. M. Caruso, "The Business Case for Software Reuse," *IBM Systems Journal* **32**, No. 4, 567–594 (1993, this issue).
9. G. Booch, *Software Engineering with Ada*, Benjamin Cummings, Menlo Park, CA (1987).
10. STARS, *Repository Guidelines and Standards for the Software Technology for Adaptable, Reliable Systems (STARS) Program*, IBM CDRL No. 0460, STARS Technology Center, Affiliates Desk, Suite 400, 801 N. Randolph Street, Arlington, VA 22203 (March 15, 1989).
11. J. Nielsen, *Hypertext and Hypermedia*, Academic Press, Inc., New York (1990).
12. W. B. Frakes, "Software Reuse, Quality, and Productivity," *Proceedings of the International Software Quality Exchange '92*, Juran Institute, Inc., San Francisco, CA (1992), pp. 9-9 to 9-18.
13. R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software* **4**, No. 1, 6–16 (January 1987).
14. E. Karlsson, S. Sivert, and E. Tryggeseth, "Classification of Object-Oriented Components for Reuse," *Proceedings of TOOLS'7*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1992), pp. 1–13.
15. RIG Technical Committee on Asset Exchange Interfaces, "A Basic Interoperability Data Model for Reuse Libraries (BIDM)," Reuse Interoperability Group (RIG) Proposed Standard RPS-0001, April 1, 1993. Note: The Reuse library Interoperability Group is a group of government, industry, and academic participants interested in the development of interoperability solutions. Their material is available from AdaNET (telephone 800-444-1458) and ASSET (telephone 304-594-3954), or RIG Secretariat, c/o Applied Expertise, 1925 North Lynn Street, Arlington, VA 22209.
16. R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," *Communications of the ACM* **34**, No. 5, 88–97 (May 1991).
17. K. Laitinen, "Document Classification for Software Quality Systems," *ACM Software Engineering Notes* **17**, No. 4, 32–39 (October 1992).
18. Y. S. Maarek, D. M. Berry, and G. E. Kaiser, "An Information Retrieval Approach for Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering* **17**, No. 8, 800–813 (August 1991).
19. K. P. Yglesias, "Limitations of Certification Standards in Achieving Successful Parts Retrieval," *Proceedings of the 5th International Workshop on Software Reuse*, Palo Alto, CA (October 26–29, 1992), pp. YGL 1–5.

K. P. Yglesias
IBM Large Scale Computing Division
Poughkeepsie
New York



A reusable parts center

In 1991 the Reuse Technology Support Center was established to coordinate and manage the reuse activities within IBM. One component of the established reuse organization was a Reusable Parts Technology Center in Böblingen, Germany, with the mission to develop reusable software parts and to advance the state-of-the-art in software reuse.

The history of the Böblingen parts center dates back to 1981. It started as an advanced technology project looking at methods for reusable design. A recent activity was to develop a comprehensive class library for C++**. This library is offered together with an IBM product (IBM C Set ++ compiler).

In the beginning the goal of the project was to have an integrated software development system that supported the reuse of parts. A first approach tried to find appropriate parts by analyzing existing code, but lead to the belief that parts of code can easily be reused if they are realizations of abstract data types. Projects that followed verified this, and now the parts center offers implementations of abstract data types for different languages and operating systems. This entry in the forum describes how the parts center evolved and what experiences were gained by this effort.

The need for a parts center. Before the existence of a reusable parts center, reuse of code across project borders seldom took place. No organizational structures supported cross-project communication. In addition, the lack of a common design language made communication difficult. Many different description methods for code were in