BUILDING AND MANAGING SOFTWARE LIBRARIES

Gerald Jones, Ruben Prieto-Diaz

Computer and Intelligent Systems Laboratory
GTE Laboratories Incorporated
40 Sylvan Road
Waltham, MA 02254

ABSTRACT
There are numerous issues (both technical and non-technical) involved in establishing a successful software reusability program within an organization. A necessary first step is the ability to organize and catalog collections of software components and provide the means for developers to quickly search a collection to identify candidates for potential reuse. In addition to the technical issues of component classification and library retrieval, there are many important managerial and organizational issues involved, such as standards, training, maintenance, incentives, procurement, certification, user follow-up, and version control.

In this paper we present an overview to building and managing software library systems. We first present material on software classification--surveying existing techniques, describing our research on faceted classification, contrasting the different approaches, and describing the process of constructing a faceted classification scheme. Next we describe an automated library system based on faceted classification that is currently being prototyped. Then we describe some management issues related to establishing a reusability program within an organization and effectively using a software component library.

SOFTWARE CLASSIFICATION

Survey of Existing Techniques
The majority of existing software libraries, both old and new, use some form of a hierarchical, enumerative scheme as the primary method of component classification. This is a predefined listing of all possible component categories hierarchically arranged with each component being assigned to one of the categories.

One of the first classification schemes for computer programs was that introduced by the IBM Users Group (SHARE) in the early 60s [4]. This scheme is a three level hierarchical scheme. A modification to this scheme was proposed by Bolstad [5] that included 22 first level classes and as many as six levels for some of the classes. Bolstad's scheme has served as the basis for several other collections, such as the Guide to Available Mathematical Software (GAMS) [6], and the International Mathematics and Scientific Library (IMSL) [7].

These schemes emphasize the mathematical software categories. A more general 'computing' set of categories was introduced by the Computing Reviews of the ACM in 1960 [8]. Listing 33 computer related categories with no subdivisions (one level hierarchy). The latest revision of this scheme in 1982 [9] now includes a three level hierarchy with 11 first level categories. This scheme is being used as the basis for a software library system at the NASA/Ames Research Center [1].

Other examples of enumerative schemes for software libraries include Intermetrics' Reusable Software Library (RSL) [10] which
uses a predefined hierarchical category code for describing component functionality. NASA’s COSMIC Software Catalog [11] which uses a flat list of 75 NASA subject categories for classifying its programs, and Toshiba’s PROMISS system [3] which uses a list of function codes for categorizing its registered programs.

In our research at GTE Laboratories, we have proposed the use of faceted classification for reusable software libraries [12,13]. Faceted classification is an important area of study in library science and faceted schemes are used in many libraries in India and Europe. Unlike the enumerative approach which relies on the breakdown of the universe of knowledge into a predefined hierarchy of standard classes, the faceted approach relies on synthesizing new classes from a set of independent, elemental classes called facets. Facets are considered the perspectives, viewpoints or dimensions of a particular collection or domain.

For example, the faceted classification scheme developed for the software ‘asset’ collection at GTE Data Services (GTEDS) currently uses five facets: Functional Area, Action, Object, Language, and Hardware. The Functional Area is the general functional name (e.g., keyboard interface or file handler) for the area in which the component resides. Action and Object are the lower level verb/noun descriptions of the function performed by the components (e.g., format/reports or convert/dates). An individual component may have multiple action/object descriptors. The Language facet indicates the programming language used to write the software component, and the Hardware facet indicates the machine environment for the component (e.g., IBM Mainframe, PC or Tandem).

In a faceted scheme, a component is classified by selecting the most appropriate term (or terms) from each facet to best describe the component. This results in a new, synthesized class that is tailored to the individual component. For example, (keyboard interface/input/strings/PC) would be a possible component descriptor for the GTEDS collection. The standard set of terms for each facet are ordered according to how closely related they are to each other. This important feature provides a means to search for similar components within a collection.

In addition to some basic classification scheme, many software libraries also include other mechanisms for helping locate components. Several systems allow lists of keywords to be attached to each component. In the COSMIC catalog, for example, an average of six keywords is assigned to each component. These keywords come from a controlled source—the NASA Thesaurus of about 3,000 entries. In other systems, such as Intermetrics’ RSL and the NASA/Ames system, arbitrary programmer-defined keywords can be attached to each component. Keywords serve as extra keys for searching for library components, usually to supplement the basic classification mechanism.

Many library systems also allow other attributes of the component descriptor to be used for retrieval. The COSMIC catalog, for example, is indexed by the program author and program identifier in addition to the basic category code and keyword list. Intermetrics’ RSL uses a list of 14 component attributes (such as unit name, machine, compiler, author, date created, etc.) that can be used for retrieval in addition to the category code and keyword list. The current version of our prototype library system allows the title and descriptor fields of the components to be searched for an arbitrary string in addition to the faceted term searching.

One example of a library system that does not use either an enumerative or a faceted classification scheme is Bell Labs’ CATALOG system [14]. This system is based on information retrieval (IR) techniques for locating components in the library. Either the component itself or an associated component descriptor is indexed by the IR system, so that each (non-trivial) word can be retrieved in a search. There is no manual classification of the components into pre-defined categories. A component descriptor template is suggested that includes programmer defined keywords, descriptive text, and many of the same attributes described above (author, date, environment, etc.). A user can search for an arbitrary word, phrase, or fragment within specific fields or across the component or descriptor as a whole. Further comparison of the information retrieval approach with the faceted classification approach is provided in the next section.

Comparison of Approaches

An inherent problem with enumerative schemes is traversing the hierarchical tree to find the appropriate class. In the Dewey Decimal system [15], for example, the title Structured System Programming could be classified in any of the following classes: system analysis (001.81), software (001.642 5), systems (003), systems analysis (620.72) or systems construction (620.73). To compensate for such ambiguity cross references are established, but cross referencing is a cumbersome and error prone process.

Rather than forcing the librarian to choose from a single hierarchy of standard classes, faceted classification synthesizes a new class, tailored to the particular component, from the elemental facet terms. In the GTEDS classification scheme, for example, the class

```
keyboard-interface/input/strings/C/PC
```

was synthesized using terms from the five facets and may only be applicable to one component.

In addition to providing more precision than the enumerative approach, the faceted scheme also provides better expandability and flexibility. In a faceted scheme, new classes can be easily added as new terms under the appropriate facet, and new facets can be added if the collection grows into a new dimension. By contrast, in an enumerative scheme, adding a new class may force rearrangement of the hierarchy or at least the generation of several new cross references from related classes.

Perhaps the most important advantage of the faceted approach is that it allows multiple dimensions to be defined to aid retrieval of components. The enumerative approach defines a single hierarchy of terms that is used to classify components in one dimension, usually the broad functional area for the component. The faceted approach, however, allows multiple dimensions to be defined and used to classify the components. For software components, for example, in addition to some basic facets to describe a component’s functionality, facets could be defined for the computer environment, the granularity, or the structure of the component. This allows the user, then, to more precisely formulate his query and retrieve components that more closely match his needs. This is especially valuable with large collections where simple browsing of the hierarchy is not feasible.

Both the faceted and enumerative schemes rely on a controlled, standard vocabulary for classifying components in a collection.
In faceted classification, each component is assigned the most appropriate term (or terms) from the standard list of terms within each facet as its classification descriptor. A user query is then searched against the set of component descriptors.

This is quite different from the information retrieval (IR) approach exemplified by the CATALOG system. For this system, the component itself or an existing text descriptor for the component is automatically indexed to provide search capability for each (non-trivial) word that occurs. No manual classification of the individual components is done.

There are advantages and disadvantages to both of these approaches. Clearly, the advantage of the IR approach is that the manual process of creating and maintaining the classification vocabulary and cataloging each component in terms of the vocabulary is eliminated. The overhead cost for this manual process is significant, though experience and a better understanding of the process (see the next section) help reduce the overhead.

One key advantage of the faceted approach, on the other hand, is that a standard vocabulary is provided. The same standard set of terms is used to both classify and retrieve the components, thus enhancing retrieval effectiveness. In the IR approach, if different developers used different terminology in describing their components (a frequent problem), users may not find all the appropriate components. Terminology differences are the source of major problems in the computer field, and the existence of a standard vocabulary for describing components is a big help, especially for libraries that spread across large groups and different organizations.

Another major advantage is that similarity of terms within the standard vocabulary can be established. This allows users to retrieve groups of closely related components. If the terms used in a query do not yield many 'hits', some similar terms can be suggested to the user to broaden the search. This is not possible with an automatic indexing scheme. The faceted scheme allows searching based on semantic similarities, rather than just syntactic word similarities.

Another disadvantage of the IR approach is that it presumes the existence of high-quality text descriptors for the components. If text descriptors do not exist or are incomplete or out-of-date, then supplemental descriptors will have to be written, but this defeats the main advantage of the IR approach.

Our view is that both these approaches should be exploited to complement each other. For the major search criteria to be used for the collection, a faceted classification schedule should be developed, standardizing the vocabulary for the collection and providing the added semantic similarity information that greatly aids the retrieval process. For all the additional textual information that is available for the components, an IR search mechanism should be used to supplement the retrieval process and allow for any unanticipated type of query. We have included a simple text searching mechanism to our prototype library system, and we have plans to expand its functionality in the future.

Construction of Faceted Schemes

As mentioned above, a drawback in faceted classification is the overhead cost in developing, using, and maintaining the classification vocabulary. To overcome this problem we have concentrated some of our research efforts on how to simplify these processes. Although not yet automated, we have developed a methodology and a set of guidelines to help the librarian or organization in constructing a faceted classification scheme for a software library.

To illustrate the methodology, we present a simple yet comprehensive example drawn from library science. The specialized faceted classification schemes of library science are based on a process called literary warrant. Literary warrant consists in selecting a random sample of titles from the collection to be classified, listing individual terms from the titles, grouping related terms into common classes, and organizing the common classes into a classification scheme. This process implies knowledge of the domain and of the intended use of the collection.

Assume we are asked to build a classification for a list of zoology related titles (i.e., books). The first step is to select a representative sample from the collection. Let us assume we select the following titles:

- "Essays on the physiology of marine fauna"
- "Animals of the mountains"
- "Amphibious animals"
- "Desert reptiles"
- "Migratory birds"
- "Salt water fish"
- "Mammalian reproduction"
- "Snakes of the Amazon river"
- "Experimental reports on the respiration of vertebrates"
- "Tropical leaf moths"

The next step is to list the individual terms from the titles (or abstracts) and attempt to group terms sharing some attribute. This is basically a clustering exercise. An initial set of groups is:

- physiology, reproduction, respiration
- tropical, desert, mountains, salt water
- marine, amphibious
- fauna, animals, vertebrates, reptiles, snakes, birds, fish, moths, mammals
- essays, experimental reports

Identification of groups is carried on concurrently with clustering. The idea of a group or class must be present before we can group terms under it. Five facets are identified for this example: by processes, by habitat, by element, by taxonomy, and by literary form. The groups are then ordered by their relevance to the collection's users, and terms in each group are listed in a logical order. The resulting classification scheme is shown in Figure 1.

It is assumed in this example that the users are ecologists, and, therefore, habitat is the most relevant facet and is listed first. Terms within each facet are arbitrarily ordered to show some logical relationship among them. For example, in 'taxonomy' the order is from general to specific.

After an initial classification scheme is developed from this analysis of a representative sample of the collection, then the rest of the collection is analyzed to see if their terms fit in the existing facets. Terms that do not fit are then clustered to identify additional facets that were not discovered in the initial sample. After further facet reordering and refinement of terms, a final scheme is listed. We can now systematically use this scheme to classify each title of the collection by selecting a
DOMAIN -> animals/fauna

<table>
<thead>
<tr>
<th>(by habitat)</th>
<th>(by elements)</th>
<th>(by taxonomy)</th>
<th>(by process)</th>
<th>(by literary form)</th>
</tr>
</thead>
<tbody>
<tr>
<td>land</td>
<td>marine</td>
<td>animals/fauna</td>
<td>physiology</td>
<td>essays</td>
</tr>
<tr>
<td>tropic</td>
<td>invertebrates</td>
<td>invertebrates</td>
<td>respiration</td>
<td></td>
</tr>
<tr>
<td>desert</td>
<td>aquatic</td>
<td>invertebrates</td>
<td>reproduction</td>
<td></td>
</tr>
<tr>
<td>mountain</td>
<td>amphibious</td>
<td>moths</td>
<td>reports</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vertebrates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>water</td>
<td></td>
<td>mammals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sea</td>
<td></td>
<td>birds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>river</td>
<td></td>
<td>reptiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>snakes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon</td>
<td></td>
<td>fish</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1 - Faceted Classification Scheme for Example

Combination of terms, one from each facet, that best describes the title. For example:

Essays on the physiology of marine fauna
  ➔ //marine/animals/physiology/essays

Animals of the mountains
  ➔ //mountain/animals/

Desert reptiles
  ➔ //desert/reptiles/

The tradeoff between simplicity and generality of the scheme must be kept in mind while going through this process. On the one side we do not want to have an excessive number of facets that will make the scheme difficult to use and on the other side, we do not want to have so few facets that the classification becomes too general.

Although apparently simple in this example, naming the facets may be a difficult task. Generic and unambiguous names should be selected. Since facets are theoretically orthogonal in the conceptual plane, each facet name must be clearly distinct from the others.

As new titles enter the collection, new facets may be defined and new terms added to the scheme. This is briefly illustrated, the core of literary warrant. Buchanan [16] and Vickery [17] present detailed tutorials of this approach.

One important point that should be emphasized is that a facet classification scheme needs to be tailored to a particular collection, domain, and/or organization. In order to have effective retrieval of relevant components, it is necessary to use the perspectives and terminology appropriate to the collection and the organization. There is great variety in software collections and in component information available and needed by organizations. It is not reasonable to think that some standard set of facets and computer science terms will suffice for classifying all software collections.

It may also be necessary to use several different classification schemes within the library system for one organization. For example, we are working with one GT organization that is interested in classifying several different collections—system software code components, hardware descriptors, training documents, and personnel skill information. Each of these collections will require a different set of facets and vocabulary terms for their classification. Even two software collections within an organization may require different facets and terms to represent different domains and intended uses.

The development of the classification scheme(s) for an organization is a significant activity that must be budgeted, but it should not be an overwhelming task. The guidelines described above should provide considerable help. Consultation with project groups like GTE Labs that have experience doing software classification is also very helpful. And the classification schemes can be quite simple in many cases. For a software code collection, for example, one facet to describe the functionality of the component and one facet to indicate the environment needed to use the component may be all that is needed. As the collection grows, other facets that will help discriminate the components in the collection will likely become obvious and can be added later.

We are currently working on more detailed guidelines for building faceted classification schemes, as well as guidelines for doing component classification, for creating thesauri for the facets, for developing conceptual graphs for measuring term closeness, and for updating the scheme as the collection grows and changes. We are also performing related research in the area of domain analysis. Whereas the classification guidelines described above focus on building a classification scheme for existing collections, domain analysis focuses on a top-down analysis of an application domain. One output of the domain analysis process is the development of a general classification vocabulary for the domain that is not biased toward a particular collection of components.

A LIBRARY SYSTEM BASED ON FACETED CLASSIFICATION

A prototype library system based on faceted classification is currently being developed on the IBM-PC by the authors. Figure 2 provides an overall data flow diagram for the library system, though not all the functionality is currently implemented (see below). The system contains three external interfaces—the user, the librarian, and the asset manager.
The user places requests in the form of queries to the query and retrieve subsystem and obtains descriptors of assets that satisfy the request. During the query process, the user is assisted by various vocabulary tables which contain valid terms organized in a faceted classification. A synonym dictionary (thesaurus) is used for each facet to resolve semantic ambiguities during query construction.

Closeness metrics for terms in the system vocabulary are used by the query and retrieve subsystem to provide the user with descriptors of components similar to the original request. The user can also have the system rank the descriptors by their degree of reusability. The ranking is based on reusability-related metrics such as component size, complexity and documentation quality. If satisfied, the user may then invoke the order asset subsystem to obtain an asset from the asset shelf.

The query and retrieve subsystem keeps a detailed log of all user activities in the usage data files. Off-line analysis of these files, through the analyze usage subsystem, provides the librarian and asset manager with essential feedback for customizing and tuning the system. The librarian maintains the system and has a specialized interface to access all database tables (i.e., asset catalog, classification schedules, thesauri, conceptual graphs and ranking functions).

The initial release of the system emphasizes the query and retrieve functions, the gathering and reporting of system usage data, and librarian support for adding, updating, and deleting assets. Figures 3a, 3b, 3c, 3d show some of the screens from the PC prototype for the user query function. The facets are displayed (Figure 3a) and the user can either enter terms directly or can invoke help for any facet. Facet help causes a scrollable, pop-up window to be displayed that lists all the valid terms and synonyms for that facet (Figure 3b). The user can choose one of these terms directly if he wishes.

If the user enters a term directly, the vocabulary tables are searched to check if this is a valid term for this facet. Automatic case conversion, stemming, and phonetic matching is done to facilitate this search. It is also possible to enter wild-card characters to invoke a pattern match search for a term. If a term appears in the vocabulary in multiple contexts a context clarification screen is displayed listing the contexts and allowing the user to choose the appropriate usage (Figure 3c).

The user may also enter an arbitrary string (keyword) that is used for a textual search of the component titles and text descriptors. For example, particular product or vendor names or local buzzwords that often appear in the descriptors can be used to help a search. This is a very simple implementation of IR searching mentioned above, and we plan to expand this aspect of the system in the future.
Once the user has entered the desired terms and keyword, he will invoke a search against the catalog. The system will display a scrollable, pop-up screen to list the id, title, and short description of all the components that match the query. The user may then inspect the full catalog entry for any of these components.

If the user is not satisfied with the retrieval (list too small or null), he may expand the search by selecting one or more terms of the query for expansion. This will display (Figure 3d) a scrollable list of terms similar to the term entered ordered by the closeness to the original term. The user may then select an 'expansion factor' to indicate how many terms to use for that facet in an expanded search. In Figure 3d, for example, an expansion factor of 3 would indicate to use the terms RELEASE, DUMP, and HALT in addition to ABORT when searching for a match for the ACTION facet.
Usage data is automatically captured for all user queries and retrievals. Vocabulary usage data is valuable for analyzing the classification vocabulary and tuning the facets, terms, synonyms, and term similarly information. Retrieval usage data is valuable for analyzing how the users are using the system and for analyzing the collection itself—determining what components are being searched for and what success users are having in finding what they want. Manual follow-up with the users is also crucial in this regard.

The library system is completely table-driven and is suitable for any collection of components once a faceted classification schedule has been developed for the collection. The current prototype uses two component collections for demonstration. One is the GTEDS asset collection and the other is a collection of small, UNIX code fragments. Each of these types has a different set of attributes, facets, and vocabulary terms, but only the database tables that drive the system change in moving from one collection to another. We also plan to implement a query mechanism that can search across collections, and can relate components in separate collections.

The system runs on any standard IBM-PC compatible machine and is written entirely in C. It uses the Oracle DBMS as the underlying database system and uses a windowing package called Windows for C to handle all the screen and keyboard functions. The system also is accessible from VT terminals (VT100, VT220, VT240, etc.) connected to a VAX/VMS environment using a multi-user DOS server called 386WARE from Logicraft.

MANAGING SOFTWARE LIBRARIES

In addition to the technical issues of classification and library support for a collection of reusable software components, there are many important management and organizational issues related to using such a library. Tasks related to the creation, certification, distribution, maintenance, and promotion of the library collection can happen only with a strong organizational commitment to reusability and an effective management structure to accomplish the tasks.

In this section, we present an overall view of how a reusability program could be structured within an organization to effectively accomplish these tasks. This structure is loosely based on the organization of such a program, the Asset Management Program (AMP), at GTE Data Services (GTEDS). Figure 4 provides an overall view of how the AMP is structured.

The AMP was created for the purpose of creating, maintaining, and making available a collection of reusable assets. A reusable asset is broadly defined as any facility that can be reused in the process of producing software; their initial emphasis is on reusable software components. Assets may be external (supplied from outside vendors) or internal (developed within GTE). The internal assets can be created as part of the normal product development or can be created as reusable components independent of any product development.

Overseeing the activities of the AMP is the asset management group. This group is responsible for setting the policies on asset acquisition and distribution, for managing the program budget, and for promoting asset reuse. Policies must be set to establish acceptance and certification criteria for reusable assets, to coordinate production groups on standards for software development, and to ensure the security of assets. Managing the budget includes deciding how to charge for services, keeping track of return on investment, and financial planning. Usage promotion includes incentives for developers to create and use reusable assets, advertising, and participation in design reviews to recommend use/reuse of assets.

Central to the AMP is the asset library, consisting of an asset shell (containing the assets) and an asset catalog (containing the asset descriptors). The users are software engineers from development groups with enough expertise and experience to select potential reusable assets. Typical user activities include querying the catalog, browsing asset descriptors, and placing orders for selected assets.

An asset librarians is responsible for the administration, maintenance, and expansion of the library. Administration activities include catalog (hard-copy) distribution, usage monitoring and reporting, and order processing. Maintenance activities include cataloging new assets, updating existing asset descriptors, updating the classification vocabulary, and version control of the assets. Since the catalog is the repository for all asset information, the librarian must interact with all other groups to keep catalog information up to date.

An identification and qualification group has responsibility for building the asset shell. This group identifies and certifies new assets procured either through the development groups or from external sources. Existing and new projects must be analyzed with respect to their application domain to determine which
assets should be created that would have high potential for reusability. The certification process includes assessing the component documentation for completeness and proper format, and assessing the completeness of the component test procedures. Any inadequacies in the documentation and testing must be resolved before inclusion in the library.

The help support group provides training and help for all activities related to the reusability program. They assist users in the use of the library system, and provide training for development groups on the standards and guidelines for developing reusable components.

Development and maintenance groups consist of software development teams that produce assets as assigned by the identification and qualification group or submit existing (or modified) components that they think will qualify as reusable assets. These groups may be part of product development groups or part of the AMP organization. Often the same group that submits an asset will be in charge of maintaining the asset.

SUMMARY

Software reusability does provide the potential to realize substantial productivity and quality gains in the software development process and the use of an effective software library system is an important part of a reusability program within an organization. We have surveyed some basic technical issues in the areas of software classification and library support for reusability emphasizing our work in faceted classification that we believe provides a solid basis for building and maintaining large, wide-spectrum software library systems.

Reusability will not happen by itself, however, by simply building a reusable component library. There must be ongoing management support of the form described above to operate a reusability program with the resources and authority it needs and provide the overall atmosphere that encourages and emphasizes reusability throughout the development process.

REFERENCES


