Progress in Reusable Parts Libraries (or Lack Thereof)

Kathryn P. Yglesias

IBM Corporation
Mailstop P419/921
93 Myers Corner Road
Wappingers Falls, NY 12590
Tel: (914) 296-6587
Email: yglesias@vnet.ibm.com
Fax: (914) 296-6496

Abstract

Since Booch's abstract data type parts library was published in 1987, many part libraries have become available but coverage of the domains of software remains minimal. There are dozens of abstract data type (ADT) and graphical user interface (GUI) libraries. There are also some libraries available to meet selected markets, especially when the libraries are built to complement a consulting/service offering, for example those offered by Price Waterhouse or KPMG Peat Marwick. It is difficult to ascertain why a broader range of general purpose libraries and more domain-specific libraries have not been built. Some experiences with information and software libraries indicate technology limitations and technical change as two contributors.

Keywords: reuse, reusable parts, part libraries

Workshop Goals: uncover reasons for redundancy in libraries and inhibitors to broadening range of parts

Working Groups: all

1 Background

Kathryn Yglesias has worked with the IBM Reuse Technology Support Center for four years. Her areas of focus have included defining classification schemes for IBM parts; evaluation of retrieval success; exploration of applying software reuse concepts to other software life cycle artifacts such as process documents, user information, and designs; consultation; and planning consultation offerings and writing standards manuals.

2 Position

2.1 Current Parts Libraries

In the information area, much of the "reusable information" is in the form of CD-ROMs. The type information being successfully marketed includes various versions of the Bible, dictionaries and encyclopedias, and other information typically found in the Reference section of libraries. The templates for legal forms also have a wide audience and are more similar to software reusable parts.

Software has the ever more abundant ADT and GUI libraries. These are being packaged with compilers, debuggers, builders, and other associated aids frequently now. These support tools respond to the programmers' need to be able to quickly understand what a parts set (library) contains and help using it. This is an important development in institutionalizing reuse.

The current set of parts libraries which are publically available can be summarized as containing many (but not all) of the basics in a wide set of variations, but containing little diversity. The IBM internal set of libraries also includes multiple variations of the standards, as well as libraries specific to software segments or niches. A subset of the specialized parts are reusable outside of originator's domain or product group.

2.2 Problems Encountered

Initial attempts to build parts libraries were based on a trial and error process. Many things were tried, then iterations were made to modify the sources inhibiting success. This feedback mechanism benefitted the developers of our most successful parts libraries of today. In the non-code area, proposal and process libraries were built. The proposal library taught us that having the real experts take time to write the generic, reusable information is the only way to be successful. It is always hard to get these people's time, but it is a truism that they are needed. Although experts were used, some of the more technical areas written for reuse were not a reuse success. It was found that the technology changed so rapidly that the extra cost to make the information reusable could not be recovered before the information was no longer timely. Another problem with reusing very technical information is that the frameworks (customer environment, terminology, etc.) change so much that extensive editing is required for reuse. Templates have proven to be the most common components in our information libraries.

In the software/code area, some of the problems which have led to disuse of parts libraries are change in language focus. For example, in IBM the effort to make applications portable to our many platforms led to changing from PL-based languages to C or C++. The successful PL libraries were either ported or have a declining usage. Another problem area associated with languages is the "good" compiler "lag". By this I refer to the many Ada programs which were written and then re-compiled and re-tested many, many times as the compiler problems were gradually resolved. This type problem continues to occur, but to a lesser degree. However, the overhead of re-testing, and sometimes, modifying a large set of components for every compiler release led some (especially Ada) programmers to abandon reuse.

A newer generation of reuse problems is occuring for OO programmers as they try to reuse class libraries built by other product groups. This class of problem offers some interesting challenges for the next few years.

2.3 Creating Reusable Parts Libraries

The parts in a part library have been found to be more widely used when the parts form a collection, that is, they have common structures and purpose. Libraries of this type may be called "architected", which is defined as a set of related parts which (ideally) have the characteristics described below. [1] Experience has shown that architected parts sets are more frequently reused than unrelated collections of parts. A contributing factor is that these characteristics also reflect the goals of good software engineering. And once the reuser becomes familiar with the structure of the library, it is easier to know how to use any part in the library and to guess whether a needed part is likely to exist within the library's domain.

The characteristics of these planned libraries show that they are successful because

1. the parts individually meet the technical requirements for reuse in terms of:
   - Complexity (hiding of)
   - Adaptability
   - Configurability
   - Predictability (the level of fidelity of meeting performance and resource requirements)
2. the parts are structured to work together in the following ways:
   - Extensibility (ability to extend the domain of the architecture)
   - Scaleability (ability of architecture to map to implementation of increasing dimension within the domain)
   - Functional composability (ability of components to be combined with other components in the architecture)
   - Interoperability (ability of components to be integrated with other software not in the architecture)
3. and user considerations are met in the following ways:
   - Understandability
   - Usability
   - Quality (of components and supporting documentation)
   - Compatibility (with existing standards, terminology, and technology)
   - Salability (ability of architecture to reflect the requirements in the eyes of the end-user).

These characteristics can be used for software (code) and information, although some of the terms are different than those typically used to measure "goodness" in information. When the list is read, terms such as, "understandability", "usability", and "quality" apply directly to information. However, a term such such as "consistency" might be used in place of "predictability".

The architected libraries which we have had for several years are ADTs, GUIs, graphics objects and proposal text segments. These libraries have sustained high usage and high acceptance. It is my belief that an understanding of the characteristics of architected libraries (and the domains) by the developers led to the positive reuse results of these libraries.

2.4 Reusability and Your Goals

Before continuing, it may be beneficial to remember why reusability, architected libraries, and domain analysis are being discussed. The purpose is to use an understanding of an organization's business objectives and reuse principles to define what parts (architected sets) will most benefit the organization by their creation and reuse.

The methods described here have been used on software projects to achieve significant cost savings over traditional software development methods. Informal use of these methods has resulted in similar savings in the documentation arena. A cost analysis should always be performed before doing the domain analysis and producing the architected libraries.

2.5 Conclusions

My experiences indicate that it is possible to increase reuse with an increased number of libraries, and that many of the libraries need to be domain specific. To successfully build domain specific libraries, consideration of the following increases the chances for success:

- Know your business objectives
- Understand reuse principles
- Do a domain analysis
- Avoid leading edge technology areas or those with a high rate of technical change
- Control the risks associated with evolving technology (e.g., language changes).

3 Comparison

Much effort in programs such as STARS has been expended towards understanding how to capture domain knowledge and "encapsulate" it. [2] This usually includes domain analysis which drives the creation of domain specific parts stored in a reuse library. This approach is worthwhile, but the visible results of such an approach are seen only in the government arena.

Some observations can be made when contrasting the number of parts available commercially and through the government. First, in the commercial arena there must be a large enough audience who understands reuse, is willing to use "someone else's" software, and is in the selected domain to justify taking the part (set) to market. A second observation is that the government does not have this constraint. A third observation is that any government organization can define its domains without regard to how other organizations divide theirs, that is, there is no predefined need for the guidance systems domain, for example, of one military organization to be the same as for another. The consequences of these observations are that (1) it is possible for any single government organization to make progress in their "domain", and (2) if they had to first get agreement on their domain (to establish a "market") with all other government organizations in the same general business, then there would not be many parts in government libraries either.

My conclusions are that there is not enough concensus on "domains" across the software industry for other part libraries to be developed. If there are not consistently recognized domains or segments, then an insufficient market will exist. The problem is similar to making use of ADTs – unless the programmer has been trained to understand and look for opportunities to use ADTs, the need for ADTs does not exist.

My position is that it is necessary to expand the scope of our available parts libraries in an architected manner, along domain lines. This must be done if we are to sustain progress in institutionalizing reuse. Therefore, emphasis must be placed on defining "domains" and getting concensus on their definitions.

References

[1] "IBM Reuse Methodology: Domain Analysis," tech. rep., IBM Reuse Technology Support Center, February 1992.

[2] R. Prieto-Diaz, "Reuse Library Process Model," Tech. Rep. AD-B157091, IBM CDRL 03041-002, STARS, July 1991.

4 Biography

Kathryn P. Yglesias is an advisory systems analyst in the IBM Reuse Technology Support Center where her work includes information model definition, classification evolution, and requirements definition for corporate standards and tools. She coordinated the IBM definition of formal methods for reusing non-code work products, especially customer documentation. Her experiences include engineering and project management for the Space Shuttle program and customer liaison for an internal computer systems organization. She is a member of the AIAA and Society for Software Quality.