JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2002**

Vol. 1, No. 1, May-June 2002

The Deplorable State of Class Libraries

Dave Thomas, Bedarra Corporation, Carleton University and University of Queensland

1 THE DEPLORABLE STATE OF CLASS LIBRARIES

The OO community is full of rhetoric on the benefits of robust, portable class libraries as the substrate for building frameworks and components. Sadly the reality is far from the promise. We are long on talk but come up short on delivery.

Consider the foundation class libraries for example. It is almost universally accepted that a foundation class library is one of the critical success factors in building better quality software on a reduced time scale. Reuse measurements show that the largest technical contributor to reuse is the base class library.

Despite their importance, it is a well-known fact that these libraries receive very little development resources from library producers or consumers. The only exceptions are some libraries where there is substantial investment focused on surfacing features of a particular platform rather than on the more generic library elements.

Most libraries are spontaneous acts of creation developed quickly as part of a much more substantive product/platform development effort. In few, if any, cases are they specified, designed, implemented and tested using best practices. Most are debugged by the code that sits above them rather than by using unit tests etc. This results in a large amount of accidental complexity, code bulk and associated defects.

While there are well known algorithms with superior time and space, the rush to get libraries out the door often means using less than optimal algorithms. We should be striving to build robust libraries of the quality of the numerical analysis libraries such as NAG.

The Sun Java classes are unfortunate but wildly used examples of spontaneous creation, which continue to be plagued with quality problems. The Java community knows and acknowledges this, but the problems continue with no solution close at hand. The highly touted JCK compliance tests have numerous cases that test for the "correctness" of the known behavior of the vendor's implementation, rather than external specification of the behavior. The compliance tests therefore discourage independent

Cite this column as follows: Dave Thomas: *The deplorable State of Class Libraries*, in Journal of Object Technology, vol. 1, no. 1, May-June 2002, pages 21-27, http://www.jot.fm/issues/issue_2002_05/column2

implementation of improved time and space versions since they will of necessity differ in internal behavior from those of the vendor. Everyone would benefit if compliance test suites were in the hands of a benevolent.org rather than the vendor.

It seems that when vendors replace the original libraries, they are unable to show restraint and fix the original problems. This happens because they build on the original poor code/design. But often it seems to be Fred Brooks' "second system effect", where developers are too ambitious and attempt to cater to every possible feature and option. This increases the code bulk and reduces usability and maintainability. Feature creep begets complexity that in turn begets code bulk and increased maintenance. We need to discipline and reward developers for writing less code, ideally the smallest amount that is needed.

Current libraries are unfortunately monolithic and highly coupled. In a world, which is rapidly moving to incremental web based distribution, the tight coupling between unrelated classes can be problematic. One need only look at how difficult it is to remove the unnecessary goop from a Java or Smalltalk program to package it as a standalone or embedded application, to understand the magnitude of the problem. We need more focus on modularity and composability in library design!

The lack of external behavioral specifications, test suites and illustrative examples means that customers have little if any ability to receive and apply incremental fix/feature releases of selected library elements. The situation is further exacerbated with a lack of accepted practice for maintaining base class code in the presence of customer or third party fixes and enhancements. We need to give much more consideration to the use of interfaces vs. frameworks. Experience has shown that much more care and expertise is needed to develop and maintain closely coupled frameworks (i.e. fragile superclass problem). API style interfaces on final classes are usually much easier to maintain and provide a stable public view that can be counted on.

2 STANDARDIZATION ISN'T WORKING

Unfortunately, standardization efforts typically start with a language, or a platform API and ignore or at least defer the standardization of the library. Few seem to appreciate that the library is itself a major extension of the language and program portability and correctness depend as much on the library as the underlying language.

We have relatively mature specification technology and specification experience in programming languages but very little of this experience has made its way into library standardization. The IEEE Posix standard at one point for example omitted specification of the return codes making it impossible today to develop a multi-vendor file browser because the return codes for just the file and directory APIs are different across compliant implementations.

The reality is that today's libraries are standardized by their mass, rather than by their specifications. Typical libraries are defined by what is euphemistically called an “operational specification" superficially expressed by class or interface APIs. In practice the vendor says here is what I have and if you want a standard take it or leave it. Without proper compliance test suites not even the creator of the library can make accurate statements about it. Unfortunately test suites are so detailed and difficult that most standards leave such detail as an "exercise for the student".

3 WHY DON'T LIBRARY PROVIDERS MAKE MORE OF AN INVESTMENT?

One might well ask if Smalltalk, C++, and Java libraries are so important why are the vendors investing so little in them? First, building class libraries with their inherent frameworks is a time consuming and specialized skill. Second and most importantly these highly skilled resources are applied in other important areas such as IDEs, compiler, and runtime (vm, gc) development. Third, and perhaps most telling is that customers currently buy languages or platforms, which have the base classes bundled into the product.

Bundling of libraries with IDE, compiler or VM creates defacto standards and control points for language/platform vendors. Look closely and you will see lawyers from the same companies arguing out of both sides of their mouths on whether APIs can or cannot be protected under structure and sequence legalities.

Both the legalities and the bundling, effectively eliminate the presence of commercial third party libraries. This bundling means it is difficult for a third party to have an impact unless they have a great deal of momentum. Further, many third party libraries have all of the same problems as vendor developed libraries since they are not held to any demanding standards.

4 WHY NOT LEAVE THIS TO THE RESEARCH COMMUNITY?

The academic community doesn't seem to consider creating good class libraries a research problem even in a software engineering department. While there are many very sophisticated algorithms published, there is little interest in or emphasis on the class library design and implementation. We applaud an elegant proof, a great UI, but do not seem to see merit in a well-designed and documented library. Those that are produced are usually the accidental side effect of another endeavor. To be fair, the academic community lacks the customer perspective and can easily lose sight of real world constraints.

There are some critical problems where R&D effort is clearly needed. This includes things like language and vendor independence; currently we rely on a single implementation of a given class, which is ported to the vendor/customer platform. Given the obvious linkage between language design and library design it is surprising to see so few researchers concerned with the problem.

5 OPEN SOURCE TO THE RESCUE?

This seems like a wonderful opportunity for a community minded effort to develop an open source class library. Many experienced library developers feel it takes a small focused team to build good libraries. The intricacies and interaction in library design seem to challenge even the most disciplined teams and seem beyond the reach of all but the most disciplined open source development processes. The GNU Class path effort will be an interesting test case.

While there are many outstanding open source contributions there is naturally a much larger number of poor quality implementations. The Open Source GUI libraries such as Motif and GTK are both problematic and are in large part responsible for the lack of Linux use on the desktop.

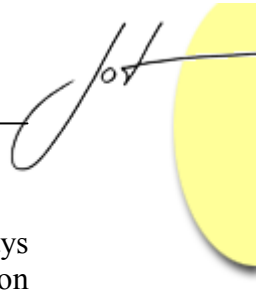
Class libraries are not considered exciting, challenging so there is a natural tendency for people to move to the top of the open source pile rather than crawl to the bottom and work on drivers or class libraries. Open source hackers want to work on something glamorous and do the next new thing. Many are unwilling to make the investment to really understand a library in detail and few consider the ramifications of cross platform or cross language usability.

Recent efforts by Sun and Microsoft, while not open source, at least promise to allow others to contribute to the process of improving their libraries and test suites.

6 SO WHERE IS THE PAIN AND THE GAIN?

One could clearly conclude that if the customer isn't screaming for it, and the vendors are not investing in it then there isn't really a problem or a business need to be satisfied. The reality is that vendors make money from IDEs, VMs, compilers etc. i.e. tools and runtimes. However, both customers and vendors are suffering immense expenses to try to keep the library dirty secret quiet. There is simply too much depending on it to say the emperor has no clothes!

The lack of test suites alone makes it almost impossible to make accurate statements about quality. The longer the dirty little secret goes on, the further the goal of common infrastructure and library moves out, until eventually, it will collapse under its own weight or some major vendor initiative will take its place.



In many ways class libs have similar problems to IT infrastructure. It is always harder than it should be to find a way to manage and share the cost of common infrastructure. No single project is able to absorb the total cost; each project would actually save money if the cost could be amortized over all the beneficiaries. In practice individual projects build their own sub-optimal partial solutions for years until a visionary is able to impose corporate discipline and maturity. What we are talking about here is imposing industry discipline and maturity.

7 CLASSLIBS.COM – ONLY IN YOUR DREAMS

ClassLibs.Com is a company dedicated to producing well-engineered quality class libraries and associated compliance test suites with related services. It also provides support, training and customization as additional for fee services. Classlibs.com uses software best practices for the creation of their class libraries including design by contract; specifications of pre and post conditions; interface specifications; separation of specification and implementation inheritance; and test driven development. ClassLibs.com specification driven development process ensures that libraries are delivered for multiple language technologies.

Each library comes with examples, as well as documented time and space information. The company has contracted with leading algorithm designers to ensure that the libraries incorporate state of the art algorithms to address the needs of speed and space. While complete proof of correctness is beyond the state of the art, ClassLibs.com is working with leading research labs on proof carrying code as well as promising formal verification approaches.

The major vendors have licensed the libraries ensuring that ClassLibs.com will be able to continue to produce high quality libraries.

8 CLASSLIB.COM BUSINESS PLAN FOR DREAMERS

In order to make it clear that ClassLibs.com is an independent business it must have the optics of an independent company. Investors who see the market need for open source libraries should fund it. A properly funded initiative is essential to attract top developers to contribute their efforts and such developers will need to be well compensated to deliver on the promise.

Ideally Classlibs.com will be a virtual company using open source processes to gain access to otherwise unavailable developer talent located world-wide. ClassLib.com would have a core team who coordinate the efforts of development wizards located globally. It will use a unique incentive based payment on deliverable scheme. Classlibs.com can control the quality of deliverables and manage the cash flow.

Developers will have a secure base salary, but will make their "profit" on delivery and peer reviewed acceptance of code and test suites. Major partners may contribute an initial set of library assets and there may be some areas in which an asset should be purchased from the original developers to reduce time to market.

In an ideal world every class should have a small number of owner(s) to care and feed it. The class should also have an independent tester to represent the customer and keep the developer honest. In practice a developer and tester can comfortably own a number of closely related classes, which leads to an estimate of 3-6 people per library module. This suggests a core team of 8 -12 and a number of top individuals to develop, review and test the code. If we consider the Java libs as an example this would suggest an effort of roughly 40-60 person years.

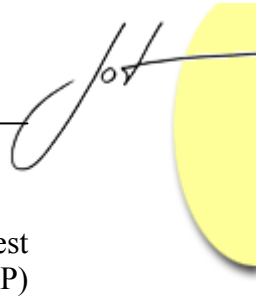
We believe that customers and vendors only tolerate the pain because they have no concrete option to obtain high quality class libraries. ClassLibs.Com will derive its revenue from subscription fees and services. The libraries will be published at regular intervals (most likely annually) with fixes as needed during the year. In addition class libs will provide compliance suites, and potentially a certification service as well as more traditional services such as education, customization and industrial quality support. Customers, including vendors, will pay for reliable supported libraries and especially for independent test cases.

9 IS THERE ANY HOPE?

It is important to note that selected researchers, vendors and users have made serious efforts to redesign and implement important class libraries. Efforts in Eiffel and Oberon have emphasized the importance of pre and post conditions in library specification and implementation as well as generics. Language vendors are now putting more effort into testing their libraries and are beginning to appreciate the level of investment required to have high quality libraries.

The C# foundation classes thankfully appears to be considerably simpler than the Java foundation classes. However, I suspect this is as much due to the experience and careful hand of Anders Hejlsberg with Pascal, Delphi and J++ than an a conversion of MFC developers to a new discipline. The ability of the .NET runtime to allow class libraries to be shared by several languages provides additional ROI for library development.

Open source efforts have made good progress in C/C++ libraries and the efforts in GNU Classpath and Mono projects hopefully will make major contributions to Java and C#. Recent commercial efforts in embedded Java libs, Java Collections hold potential for Java. Hopefully the appearance of generics in Java and C# will encourage the implementation of libraries with improved polymorphism.



Fortunately many groups in various language communities are talking about test cases. We can thank in part the test first maxim of the Extreme Programming (XP) community and supporting Junit software contributed by Erich Gamma and Kent Beck. There now exist versions of Junit for almost every popular language.

However, for the vast majority who must eat what the platform/language gives us we must daydream for the world of ClassLibs.com. Unless of course, we become much more demanding consumers and/or disciplined library producers.

About the Autor

Dave Thomas is CEO of Bedarra Corp., Adjunct Professor at Carleton University, Canada and University of Queensland, Australia, founding Director of AgileAllinace.com, and founder of Object Technology International. Bedarra works with research labs and commercial partners to transition innovations into products and practices.