LogicLibrary™

Selling Software Reuse to Management

Overview

The message from our clients is clear: CIOs everywhere are under great pressure to support new business programs, initiatives and campaigns. However, the typical enterprise IT environment is rampant with overlapping legacy applications and “spaghetti code” that is costly to maintain and impossible to adapt. At LogicLibrary, we view reuse initiatives as a powerful means not only to reduce costs, but also to support the realignment and re-factoring of enterprise development environments to meet critical business objectives.

This white paper will help with selling reuse to management in two ways. First, it will assist with the building of a quantifiable business case that unequivocally demonstrates the benefits of reuse. Second, it will explain how reuse can help transform a legacy development environment into a flexible, standardized suite of business services that can be easily and rapidly discovered and used by application developers.

The Simple Matter of a Business Case

In working to introduce reuse into the mainstream of enterprise software development, we have worked with a variety of approaches, methodologies and theories and have developed a good sense of what actually works in practice. Key among our observations is that to successfully introduce a reuse program into any organization, management must support reuse both in spirit and with the necessary resources. However, before management will allocate resources, they need to see a business case. After all, it is business decisions that must drive information technology (IT) spending in 2003.

To build a business case, you need consistent, realistic and easy-to-understand metrics. We have found that combining pragmatic and honest assumptions with a simple, easy-to-follow business model provides all the necessary evidence to sell reuse to management.

We start by asking: “How much effort could you save by reusing something rather than writing it yourself?” Typical answers span a wide range (usually from 50-100% savings) and depend heavily on factors related to your environment and existing applications. With this in mind (and based on a lot of hard data), we estimate that a reasonable savings due to reuse is approximately 80%. (The 20% effort is for locating, understanding, and integrating the reused assets.) At an industry-average cost to develop new software at around $100 per line, this means that every 1,000 lines of reused code yields a *Development Cost Avoidance* of $80,000. This calculation will surely help sell your management on reuse!

In addition, reuse allows you to avoid maintenance costs for the entire life of the product. As part of our work with enterprise reuse projects, we recommend centralized support for reusable software development assets (SDAs). This means that if a reusable component has a bug or needs enhancements, developers who reuse the component can have it fixed for “free.” While you can estimate the reduction in maintenance costs in many different ways, we multiply a historical error rate by what it costs to fix those errors. For example, if you reuse 1000 lines of code and historically would expect to fix three errors in that code at a cost of $5,000 per error, you would avoid $15,000 in maintenance cost. The calculation of this *Service Cost Avoidance* helps further sell the concept of reuse to your management.

The sum of these two values (*Development Cost Avoidance* and *Service Cost Avoidance*) is your *Reuse Cost Avoidance (RCA). RCA* represents the total quantifiable benefits of reuse.

Next, the astute manager will ask: “How much *more* does it cost to write a component for reuse instead of for just one use?” Building reusable components requires extra effort for activities such as making a more generic design, conducting additional testing and writing useful documentation. As before, answers to this question span a wide range (usually from 0%-300% additional cost, depending on the situation) but our data shows a reasonable estimate of additional cost to lie right around 50%.

In short, the business case for reuse consists of avoiding 80% of the development costs for reusing components (plus some additional maintenance savings) minus the 50% extra it costs to build the reusable assets in the first place.

Using these metrics, it is easy to show that you break even with as little as two re-uses of an SDA! This means that if you have two related projects, it will pay to base both of them on the same foundation of reusable software assets. The result is a net savings for everyone. Even with conservative assumptions, the business case for reuse is overwhelming. As the number of related projects increases, so do the benefits. Because the metrics behind the business case take only a few minutes to explain, they make a very effective way to sell reuse to management.

Improving Your Business with Reuse

How can reuse also help your business become more flexible and productive? By assessing functional needs and requirements across a broader audience of potential consumers, the team building reusable components has an opportunity to identify commonalities, to standardize and to improve the organization in many ways. Not only do the software assets they produce address a wider range of needs, the *process* of developing these components identifies organizational best practices and raises awareness about the architectural approach that is required to pull your development activities together.

Clients of LogicLibrary have recognized that modern software architectures and infrastructures encourage a more generalized approach to software asset development. For example, both the .NET and J2EE platform architectures strongly encourage partitioning applications into tiers, such as for presentation, business logic, data and integration. Each tier provides a layer of abstraction and controlled access. Within such architectures, information stored in the data tier is accessed by components in the business tier. Business tier components perform generalized and coarse-grained functional capabilities that are, in turn, customized for specific application needs by components in the presentation tier. Components developed for one application in these architectures can often provide the same services for other applications in the architecture.

In addition, the movement towards Web services and service-oriented architectures brings with it the opportunity to create generally reusable coarse-grained services that are accessible through industry-standard Internet protocols. A service-oriented architecture actually requires developers to design services that have the proper granularity and generality for reuse. Because of the overhead inherent in using Internet protocols for Web services communication, these services must present large “chunks” of business function or suffer the significant performance costs associated with excessive service calls. It’s important not to fall into the trap of exposing fine-grained operations on your services. This is the equivalent to the “objects everywhere” fallacy and associated performance overhead that early CORBA developers exposed. It is this fallacy that led to the more efficient component models espoused by .NET and J2EE today. These modern component models now provide the architectures for a successful reuse initiative.

Standardizing your applications on an architecture with a common set of software components and services moves your organization towards a more flexible development model. To make this possible, LogicLibrary promotes a best practice reuse initiative that incorporates an iterative approach to asset development. When organizations attempt to “boil the ocean” by incorporating every conceivable requirement into their first reusable assets they invariably fail because it is nearly impossible to make everyone happy without losing major business benefits. Instead, we help our clients start by providing a foundational set of generally reusable core services. Next, we work with them to combine two or three narrowly scoped sets of requirements into their first business-specific reusable components.

Then, we follow up with incremental enhancements that incorporate other requirements on a priority basis. This results in the two to three reuses that allow enterprises to quickly break-even on their reuse investment, while at the same time helping lay the foundation for future reuse savings and business enhancements.

LogicLibrary Logidex

LogicLibrary offers a unique approach that uses reuse as a vehicle to revitalize your software development processes, tools and environment. As part of this change, it is crucial that you make it easy for developers to find and employ your reusable assets.

LogicLibrary Logidex is a mapping and discovery engine that represents inherently complex enterprise application environments in a graphical, intuitive way. Logidex can hold artifacts created throughout the software development lifecycle, can be used to identify and manage SDAs, and can be implemented to support reuse processes and roles – as part of any development methodology.

The Logidex SDA Library is an inventory of SDAs and their relationships to each other, the company’s business processes and the technical infrastructure. Software developers, architects and business analysts can search the SDA Library using the discovery engine, which offers capabilities ranging from simple keyword searches to sophisticated, model-based searches. Application developers can discover and employ SDAs without leaving their familiar development environment, because Logidex is tightly integrated into leading products such as WebSphere Studio Application Developer (WSAD) and Visual Studio .NET.

In addition, the preloaded .NET and J2EE assets that come with Logidex, including patterns, best practices and models, are a great way to get reuse “out of the box” within your development organization.

Summary

Implementing a successful reuse program requires significant effort, expertise and support from management. To sell the value of reuse to your management, use metrics! Start by developing your reuse business case and then conduct various “what if” scenarios with the metrics to evaluate possible outcomes. Next, consider a move to component-based software architectures and the benefits they provide. Standardizing the foundation and substance of your applications will make your development organization more flexible and productive.

Enterprise development organizations are under significant pressure to reduce costs and still deliver high-quality products. Reuse can help address these pressures and, with the right metrics, methodologies and tools, successful software reuse can be achieved today!

About the authors

Dr. Jeffrey Poulin helped lead the development of the IBM reuse measurements and return on investment (ROI) model, widely acclaimed as the premier work in this field. He currently serves as a System Architect at Lockheed Martin Systems Integration, Owego, NY. Dr. Poulin has extensive experience as the technical lead on very large systems integration projects and has authored over 70 technical publications, including a book titled *Measuring Software Reuse: Principles, Practices, and Economic Models*, published by Addison-Wesley.

Brent Carlson is vice president of technology and co-founder of LogicLibrary. Carlson is a 17-year veteran of IBM, where he served as lead architect for the WebSphere Business Components project and held numerous leadership roles on the "IBM SanFrancisco Project," a consortium of more than 100 companies united by the mission of providing a framework for Java-based application business components. Carlson is the co-author of two books: *SanFrancisco Design Patterns: Blueprints for Business Software* (with James Carey and Tim Graser) and *Framework Process Patterns: Lessons Learned Developing Application Frameworks* (with James Carey). He also holds 16 software patents, with eight more currently under evaluation.