The Business Case for Software Reuse

What IT organization wouldn't want to save money by reusing existing software development assets (SDAs) such as components and Web services? Nonetheless, reuse initiatives often flounder because of the inability to quantify the resulting ROI. In this article we'll discuss how you can build a quantifiable business case that unequivocally demonstrates the benefits of reuse. In the process, we'll also explain how reuse can help transform a legacy development environment into a flexible, standardized suite of business components and services to streamline your application development.

The Simple Matter of a Business Case

The bottom line is always king, especially in today's resource-constrained environment. Warm and fuzzy notions of "reuse is good, therefore let's do reuse" can only go so far towards getting the necessary management buy-in (along with the staffing and tools required to successfully execute a reuse initiative). To succeed, you must be able to present a compelling business case for reuse using 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 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% of the cost needed to develop the same SDA for one-time use. At an industry-average cost to develop new software of around $100 per line, this means that every 1,000 lines of reused code yields a **Development Cost Avoidance** of $80,000!

In addition, reuse allows you to avoid maintenance costs for the entire life of your product. While you can estimate your maintenance cost many ways, we do it by multiplying a historical error rate by what it costs to fix those errors. For example, if we reuse 1000 lines of code and historically would expect to fix three errors in that code at a cost of $5,000 per error, we would have a **Service Cost Avoidance** of $15,000. The sum of these two values (Development Cost Avoidance and Service Cost Avoidance) is your total **Reuse Cost Avoidance (RCA)**. RCA represents the total financial benefit of reuse, which is powerful way to sell the concept of reuse to your management.

But what about the investment needed to create reusable assets? Any CIO worth his or her salt will ask: "How much *more* does it cost to write a component for reuse instead of for just one use?" Building reusable components or services requires extra effort for activities such as incorporating broader business requirements, conducting additional testing and writing useful documentation. Again, answers to this question span a wide range (anywhere from 0%-300% additional cost, depending on the situation), but our data shows a reasonable estimate of the additional **Relative Cost of Writing for Reuse (RCWR)** to lie right around 50%.

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

Putting this all together, we get the following formula for each SDA:

(n * RCA) – RCWR = savings per SDA

where n = number of SDA reuses. If you use our recommend default values of 20% for RCA (ignoring any service cost savings) and 50% for RCWR, you have the following results for a component that would have cost $100,000 to write:

((n * ($100,000 * 80%)) – ($100,000 * 50%) = savings per SDA

Our first SDA reuse saves you $30,000 – in other words, you more than break even with the first reuse of an SDA! This means that if you have two related projects, you will have a positive ROI if you base both projects on the same foundation of reusable software!

Improving Your Business with Reuse

Teams building reusable SDAs have an opportunity to identify commonalities, to standardize processes, and to improve the organization in many ways. Assessing functional needs and requirements across a broad audience of potential consumers does more than just help produce reusable. Developing shared SDAs identifies organizational best practices and raises awareness about the architectural approach that is required to pull your development activities together.

Modern software architectures and infrastructures support a more generalized approach to software asset development. 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 reusable coarse-grained services that are accessible through industry-standard Internet protocols. A service-oriented architecture 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.

First Steps

Getting to this layered architecture usually requires 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, have your development teams focus on an initial cluster of narrowly-scoped, business-specific reusable components. These components will quickly provide a positive ROI and initial success story for your reuse program.

As your base of reusable assets grows, teams can incorporate incremental enhancements to support additional SDA reuse on a prioritized basis. This incremental approach allows you to increase the flexibility of your application portfolio where it matters most, thereby reaping business benefits such as time-to-market advantages alongside the substantial cost savings from SDA reuse.

Summary

Even with conservative assumptions, the business case for reuse is overwhelming. Deriving these benefits is best achieved by standardizing your applications on an architecture that incorporates a common set of software components and services. Because the reuse metrics in this article take only a few minutes to explain, they make a very effective way to help you sell reuse to your management and begin the move towards a more flexible and cost-effective software development model.

Sidebar: Reuse typically takes only 20% of the effort required to develop an SDA for one-time use. This effort is needed to locate, understand, and integrate the reused assets. However, there are efforts underway to reduce this cost even further. The Reusable Asset Specification (RAS) currently under consideration for adoption by OMG, along with SDA metadata repository tools that support RAS, can significantly improve your organization's efficiency. These standards and tools not only help locate and integrate SDAs but they also can provide the reuse metrics for use in the ROI calculations discussed in this article. See www.xxxxxxxxxxx.com.