

## The Hidden Cost of Software

May 29, 2003

By [Tim Chou](#)

Do you know what is the real cost of your software?

Before you reach for a calculator, be forewarned that it's a trick question. Even technology industry analysts and seasoned IT executives have difficulty putting a hard figure on the Total Cost of Ownership (TCO) of today's enterprise application solutions. What makes the equation so tricky is the sheer number of variables that must be factored in — many of which cannot be lined up in a neat column and reconciled, such as issues of security, availability, performance, problem resolution and change management.

In computing the cost of software, companies tend to focus on the purchase price and initial implementation. Yet even when fully operational, software is never *free*. It must be supported, maintained and upgraded — often across geographies and in multiple languages, currencies and regulatory climates. In fact, industry analysts estimate that the cost of maintenance equals or exceeds the cost of software, and can be as high as twice or three times the purchase price over the life of the software.

There are hidden costs as well. Owing to the resources, time and cost involved in getting an enterprise application up and running (especially globally), many corporate IT groups are hesitant to make incremental improvements or upgrades. This inertia can cost the enterprise dearly in terms of productivity, customer service quality and market competitiveness.

Without a clear understanding of the full issues and costs, businesses still are making decisions that can govern software usage for years to come. Certainly, the options open to companies are more numerous than ever before. They can outsource all of their software needs, operate their own data facilities while leasing vendor-supported software services, or build and support their own proprietary enterprise solution. Moreover, as businesses migrate to the Web, organizations now can standardize globally on a single instance of software, or integrate multi-vendors solutions into their existing legacy system.

Yet with no common basis for comparison, how can a company weigh the pros and cons of all of these alternatives?

The confusion over software TCO works to everyone's detriment: software manufacturers, systems integrators and consultants, value-added retailers (VARs) and customers. Before committing to an enterprise solution, companies should understand where their software dollars are going.

The best way to achieve greater accountability is for the software industry to create a model for assessing and measuring an organization's success in managing software. There is a successful precedent for this: the Capability Maturity Model (CMM) developed by Carnegie Mellon University's Software Engineering Institute in concert with a consortium of software manufacturers. The CMM model has become the software industry's *de facto* standard for assessing and improving the maturity of processes guiding software development, testing and implementation.

In this article, we will examine some of the hidden costs of software and the need for an industry model to help quantify, analyze and trim maintenance costs. We'll also examine how the outsourcing model offers the economy of scale to standardize and automate software processes and support specialization.

### **Software's "Hidden" Costs**

Understanding the cost of software means tracking all related expenditures across its life cycle: purchase/lease, implementation, training, support, maintenance and all subsequent upgrades. It is simpler said than done. Obfuscating the whole price issue, some systems integrators bundle the cost of the software in with their services, and quote one, all-inclusive price. For large organizations, these custom integration projects often run into millions of dollars; still, they have no idea how that lump sum is divided and expended.

How do the software costs break down?

In addressing total software related spending, JP Morgan Chase technology analyst Chuck Phillips cites a recent U.S. Department of Commerce study indicating that software license expenditures account for only 30 percent of the total. The lion's share represents labor costs, with 37 percent of spending going toward internal IT staff support and 33 percent earmarked for external consultants related to software implementations. The numbers translate to a ratio of 1:2, software license to management/labor costs; and 1.1, license fee to implementation work.

IDC recently reported that "Many companies underestimate the cost of managing applications. When you consider the annual costs of administering the applications and the cost of managing issues around performance, changes and availability, it is most likely significantly greater than the application purchase price."

When assessing "bang-for-buck," consider that software performance is less a factor of purchase price than its successful integration into the business culture. Simply put: you can invest in a world-class enterprise application solution, yet if it is poorly implemented and managed, you'll end up with under-performing, problem-riddled software.

### **Inertia vs. Innovation**

When I meet with companies to discuss their software needs, they'll typically volunteer a figure on their annual software costs. Presumably, that sum covers the whole nut, yet in reality, it is only the starting point. The real question is what are they getting for that amount.

To help frame the larger discussion, I ask corporate executives the following questions:

- When did you do your last security audit?
- Do you have a disaster recovery plan in place? If a major data center goes down, how quickly can you shift operations to an alternate location?
- Is your software delivering on its full benefits?
- Who handles major system upgrades and failures?
- Does your information infrastructure and culture support continuous innovation and change?

There are lots of hidden costs to managing software. A chief technology officer of a G2000 company recently told me that he had three database administrators (DBAs) managing 130 instances of software on a 24/7 basis. I shuddered at the thought. No matter how qualified and knowledgeable those individuals might be, three IT professionals were not sufficient for the task.

This situation is not unusual as companies try to rein in IT costs and operate "lean." Yet rather than reducing or eliminating maintenance costs, this form of economizing just shifts the expenditures around; in this case, literally out the door. While the internal staff can cope with everyday situations, their bandwidth is easily over-taxed. For heavy lifting, such as major systems upgrades and/or failures, a team of high-priced systems integrators and consultants are brought in.

The labor-intensive nature of managing software gives rise to another hidden cost: inertia. With custom legacy systems constructed like Rube Goldberg contraptions, changing one piece of software often requires reconfiguring the whole. No wonder IT groups are hesitant to risk upgrading or adding to the solution. Their motto

might as well be: "If it's not broke, don't fix it!" But technology keeps reinventing itself and making quantum leaps forward in features and functionality. In a 24/7 global economy, can any company afford to lag behind?

Increasingly companies are turning to outsourcing as a way to trim costs, concentrate on their core business and improve services. Also fueling this worldwide trend are security issues, such as site redundancy, business continuity and rapid disaster recovery.

### **Outsourcing Software Services**

According to IDC, the market for "software as a service" will reach \$24 billion a year by 2005. There is a growing body of evidence that "leaving it to the experts" results in significant IT savings and operational efficiencies. The reasons should be obvious. Independent software vendors (ISVs) do the best job at managing and supporting their own software:

- Expertise — they know the software better than anyone else.
- Economy of Scale — they can maintain, support and upgrade a large user base using universal patches, standardized processes and self-service automation.
- Brand reputation — in the highly competitive software industry, ISV's have a vested interest in continually upgrading and improving product performance and services.

As I discussed earlier, corporate IT resources can be spread fairly thin. DBAs have wide-ranging responsibilities. Let's face it: no one can be an expert in 140 different software programs. This lack of resources slows the deployment of new projects. The goal is simply to keep everything going and provide basic services. Little time is left over for innovation or service improvements. Besides which, upgrades create more work (i.e., system failures, user retraining and debugging).

In sharp contrast, large enterprise software providers can commit their worldwide resources and staff to a policy of continuous improvement. Instead of maintaining the status quo, the ISV strives to make its software services faster, better and cheaper. Owning the software you manage provides hard-to-beat advantages, such as directly linking technical support and software development groups to fix existing problems and introduce new functionality. In terms of problem resolution and change management, it is the difference between throwing a few soldiers at a problem versus an army of experts. And as the direct recipients of this focus and

specialization, the outsourcing customers profit from continual gains in security, availability and performance.

... At least theoretically, they should.

While outsourcing should be judged on service quality and performance, too often the negotiations revolve entirely on price. Some major outsourcers promise significant savings if the company turns over the entire IT operation to them. Although millions if not billions of dollars are at stake — no cost breakdowns are provided. And to realize the savings, companies are locked into multi-year contracts with stiff termination penalties. Can you anticipate your company's hardware and software needs in 2012? The analogy that comes to mind is buying a house for the asking price — with no appraisal of the physical structure or cost review of the yearly maintenance, property taxes and major renovations (e.g., the roof will need replacing in the next year or two).

This type of outsourcing deal — known as *one-offs* — also sets up a conflict of interest. Since each customer environment is unique, we're back to the costly legacy system syndrome in terms of maintenance, support and upgrades. The knowledge and innovations achieved at one customer site do not translate to the next. Therefore, maintaining all of these different set-ups is expensive, and does not bode well for the future. Like the for-profit HMO model, the scope and quality of services can eventually be compromised. Back to the same nagging issue: cost. What are you getting for those annual IT savings?

The above model also reinforces the largest historic reservation about outsourcing: the customer's perceived loss of control. Companies have different needs. Logically, they should have flexibility and a voice in how the outsourcing initiative is structured and administered. By researching their choices, customers will find that they can control as much as they want. Today, you can outsource the entire IT operation, or begin with one key business process, such as supply chain management (SCM) or worldwide human resources (HR).

With hosted services, the software can reside at the customer site, with a trusted third-party or at the vendor's data center. The outsourcer can manage the entire technology stack, including hardware and facility management; or just the software component, covering the applications, database and system management.

Some outsourcers also offer *on-demand*, or pay-as-you-go services. Like a utility, the customer pays only for actual usage. With this model, companies share the same server, creating tremendous price efficiencies.

Finally, you don't have to enter a multi-year contract. One-year outsourcing contracts are available with a 30-day cancellation clause. And while all outsourcers guarantee availability, the latest trend is to also guarantee software performance.

## **Modeling and Standardizing Maintenance Costs**

Creating a disciplined approach to software maintenance will take an industry-wide effort along the lines of The Capability Maturity Model for Software. The CMM provides a framework for organizations to move from ad hoc, often chaotic, software processes to organized knowledge-based systems that support a continuous loop of learning and innovation.

The CMM is organized into five levels of software process maturity and provides the principles and practices underlying process success.

1. Initial. At this level, few processes are defined, and success depends largely on the knowledge and resourcefulness of the IT staff:
2. Repeatable. Basic project management processes are established to track cost, schedule and functionality. As a result, the organization can repeat earlier successes on projects with similar applications.
3. Defined. The software process for both management and engineering activities is documented, standardized and integrated for use by the organization.
4. Managed. With a full set of measurements in place, both the software process and product quality are quantitatively understood and controlled.
5. Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

In conceptualizing a CMM-type standard for software maintenance, I am reminded of what works in certain outsourcing models. Real gains in efficiency and cost-savings come from models that support a process of improvement based on: 1) repetition; 2) specialization; 3) standardization and 4) automation.

As the CMM model shows, the first important step in getting a software process under control is the ability to repeat the process and arrive at consistent results. The next step is specialization. The most effective software solutions are highly granular; that is, they address the specific business needs of discrete groups of users. For instance, with an iProcurement solution, all employees use the same infrastructure and tools to order direct and/or indirect goods. However, for the software to serve a diverse corporate population, it must offer the means to control access (i.e., by title or role) and provide different levels of service. This

specialization is achieved by establishing a system of checks and controls: such as pre-authorizations, security codes and/or price caps.

Once a specialized process can be repeated, its use can then be standardized across the general user community. Standardization, in turn, facilitates to automation. Returning to the previous example: if online procurement processes are standardized, the e-business software vendor can automate usage with desktop self-service, automatic invoice approvals and routing, and extranet portals for preferred suppliers and vendors.

However, this economy of scale can be achieved only when the ISV manages its own software. One-offs do not support specialization because no two customer environments are alike. Incremental improvements at one customer site cannot be repeated, standardized or shared with other customers.

In the multi-sourcing model, the primary software vendor partners with other ISVs and systems implementers to meet the customer's full range of needs. While the primary vendor serves as the single point of contact for all customer service requests/problems, each ISV partner transparently manages its own software, typically from independent locations. With direct customer-provider contact, this solution supports the process of specialization, repeatability, standardization and automation. In fact, an improvement at one customer site can be applied to all users through a universal patch. In this way, cost-performance efficiencies are continuously incorporated into the standard software offering.

In the end, the real savings achieved when you do your job better, faster and cheaper reveal the real costs of software.