Getting 20% More With 10% Less:
A 2002 Recipe for IT Departments

Tom Love
John Wooten

Corporations have CIO's in order to provide IT visibility and culpability among the executive rank in the corporation. CIO's across the country and around the world are being asked to "reduce our IT spend by 10%" or some such double-digit percentage. This is in sharp contrast to the situation just two years ago when CIO's were asked "to scale up IT to meet the new requirements of the Internet".

The purpose of this article is to describe a new organizational model for IT departments that will allow today's CIO to achieve a 10% personnel cost reduction while delivering 20% more value to the organization.

Most CIO's think of an IT organization as a collection of systems that they support. Investment bank IT shops have groups that support their lines of business (e.g., equities, commodities, fixed income instruments, FX, asset management, etc.). With each line of business, there are people maintaining and supporting deployed systems as well as a few people building new applications or systems to support current or anticipated business. Similar organizational models based upon existing IT systems, can be found in insurance companies, manufacturing firms, retail banks and telecom companies.

An organizational model based upon existing systems is inappropriate because it does not reflect the degree of technical overlap between applications and systems built to support different lines of business within the same company. Opportunities for technical leverage are missed, resulting in a wasteful duplication of effort.

For the past 30 years there have been ongoing efforts to eliminate the unnecessary duplication of effort. However, in organization after organization, people have aligned with the specific systems they support and those people have had little incentive to seriously consider using any code developed for another system within the company.

Ongoing efforts to solve this problem by adding a new "reusability department" have generally proven unsuccessful. A more fundamental change is required.

I often refer to an important paper written by a friend of mine in the technological dark ages – 1972. Mel Conway, in his article *Datamation*, made an important observation that software assumes the structure of the organization that built the software. So if we have a Fixed Income Team and an Equity Team, we are not

likely to get a generic trading system which can be customized to support bonds and stocks. Instead we'll get a fixed income system and an equity system with little, if any, shared code. More often than not, the two systems will actually involve different programming languages and supporting technologies.

A corollary of Conway's Law was first described in my 1993 book, *Object Lessons*. It states that if you wish to change the structure of a software system, all you need to do, is to change the organization and wait a while. We can solve the reusability problem in organizations by exploiting this corollary of Conway's law. I'll describe the changes required and then describe some emerging tools to support this new structure.

IT shops have a tendency to acquire new and different technologies every time they start a new project. Let's call this *technical variety*. Technical variety is a function of the number of programming languages, report writers, database management systems, application servers, configuration management systems, persistence tools, etc. Any language or tool for which there is more than a one day training course should be included.

A result of having technical variety is that even if there is an existing service providing risk management within the organization, there is no assurance that any particular group will have the special technical skills required to maintain it. It may also be very difficult to connect modules written in one programming language with those of another or to learn a new operating system or programming language.

IT departments are expensive because of unnecessary duplication of effort – organizationally and technically. What can be done to change this unfortunate situation?

A Solution

An IT department should be structured as a set of services. These services would reflect roles within development projects – architecture, design, coding, testing, documentation, etc. Other services would reflect specialized work done for others – e.g., trading systems, risk management, trade settlement, news, ticker plants, compliance rule writing, user interface design, data dictionary development and maintenance, technical documentation, business workflow design, etc.

We've all heard ad nauseam that there is "no such thing as a silver bullet". Well I often carry a silver bullet around in my pocket to provide a tangible counter example[1]. No one thing is going to fix "an IT department". But some things do make a difference.

[1] Less often these days, given the enhanced security restrictions on airplanes and in office buildings.

Getting 20% More with 10% Less

What if a 500 person IT shop were "re-factored" into a set of 100 services provided by 3-10 person teams. Then all code in its respective stages of development would be reassigned to one or more of the 100 services teams. A team of 5 people who inherited 5 copies of a value at risk calculation in 5 different programming languages, would immediately start thinking, 'do we really need all 5? Would 3 be sufficient? Could we even get by with 2?' [2]

There will need to be a leader for each service team and each member of the team may also be accountable to a Development Leader. The Development Leader, in an HR sense, is a mentor for his or her team members. A junior architect might be paired with a seasoned architect to help develop the junior members architecture skills. The service team leader would need to be an expert in the identified service, (e.g., Risk Management).

In a matter of weeks, these service-oriented teams will have caused a remarkable amount of work to "simply go away", because it was so obviously redundant and unnecessary. Based upon my own experience helping several substantial IT departments reinvent themselves, I have found that 10% of the code maintained by any IT organization could simply go away without any loss of necessary services. An additional 10% could also go away, but it would be a harder and slower process. It could take a year or two for the second 10% in savings to materialize.

As we save 10% (50 people in this example) we reapply half of those resources to provide new functionality. The remaining 90% are still charged with reducing waste and improving the services they provide to their clients. By the time we achieve a total savings of 20%, the 25 people doing new development will have grown to 50. So we have reduced staff by 50 and applied 50 to new development with the latest available technology.

The efficiency of new development with the latest tools is far greater than using 20 year-old technologies. A Java programmer can achieve at least a 3:1 productivity advantage compared to COBOL programmers. So 50 developers are now capable of doing the work of 150 – a net gain of a person century of work per year by the end of the third year. In effect we can reduce spending by 10% and gain a 20% improvement in productivity!

Note that other business financing decisions are possible. For example, it would be possible to keep expenses flat while radically increasing service to the organization. Another option is to simply reduce expenses significantly without any new development. In a few cases cases, expenses associated with existing systems should be substantially reduced while the overall IT budget is increased due to the demand for new and different services.

---

[2] One seems far too aggressive, so let's not even consider that as a possibility.

The Services Solution Paradigm

There has been a tremendous change in the nature of the World Wide Web since its' inception. Originally created as a means to deliver static HTML pages, it has expanded as a means to access legacy systems via CGI services, to create robust J2EE compliant Web Applications that are scalable, and other essential components of the enterprise environment. None of these changes affect the nature of the web as much as the new standards that are being put into place for XML[3], WSML[4], XSLT[5], UDDI[6], and SOAP[7]. These standards will cause the redesign of the IT department because for the first time they allow organizations to build "services", functional units that do one thing and do it well, which are also capable of interacting with and using other services in standardized ways. The combination of services-based computation and portal[8] technologies for grouping and presenting the "edge services"[9] means that the IT department can build one "payroll gross to net" calculator, deploy it as a service and have every payroll system deployed by the enterprise use the same service (and incidentally get the same net pay from all of them for the same inputs[10]).

Careful design of a services-based IT structure should result in small, easy to implement, and easy to manage modules that have well defined interfaces. Simply the task of determining the interface for the service, documenting it, and deploying it via the WSML server, would provide a great benefit to the enterprise. Such systems do NOT require full knowledge and massive analyses to begin with. These changes can be made incrementally. Select one such functional unit and implement it as a service, leaving the current functionality in place. Then incrementally begin replacing references to the old functionality with requests for the new service. Select another functional unit and repeat. We often suggest that the initial service be something that can be accomplished in "100 business days", including deployment, testing, and initial integration into the enterprise. As experience is gained, the cycle can be shortened or several services can be deployed during each additional cycle. Not only has the

---

[3] XML stands for extensible Markup Language. This provides a "tagged" representation of data elements that is machine and software language independent. This is the "what" that is being described.

[4] WSML stands for Web Services Definition Language. This provides the service interface so that systems using the service know the way they must present their requests for service.

[5] XSLT stands for extensible Style Language Template. This provides the information on the "how" the information is to be presented.

[6] UDDI stands for Universal Description, Discovery, and Integration. It represents a public or private directory implementation. This provides a way to make services available without tying them to particular sites or machines.

[7] SOAP represents Simple Object Access Protocol. This is the protocol for making XML requests and responses.

[8] A Portal is a web-based presentation of information channeled through a standard provider usually in a end user configurable manner. This allows companies to provide a consistent look and feel to disparate systems and have a convenient way to present corporate information and standard services.

[9] Edge services are those services that are exposed to end users. For instance, several services might be used to carry out a NetToGross calculation, but the edge service might be named "Calculate Payroll Taxes".

[10] Surprisingly, many companies have systems in place that give differing answers for the same input data.

functional structure of the IT organization been changed to become more efficient and to reduce redundancy, but new opportunities for the expansion of the business model and delivery methods now become possible because SOAP and web services are designed to be delivery agnostic.

We would expect a typical COBOL implementation of a complex enterprise support application to be replaced by about 20% or less of the amount of the original code, be much more scalable, and be completely web-enabled. Using the incremental approach discussed above, the impact upon ongoing operations is minimized. This approach is imminently applicable to outsourcing at low costs where expertise exists in functional analysis and design in parallel with experience in building large scalable applications.

Summary

Shoulders Corporation is working with several leading edge organizations to implement a services solution paradigm including the provisioning of web services as described here. Interim results are extremely encouraging.

