# Managing New Technologies in the Organization

**Brian Goff, M. Eng., MBA**

## Background

New software technologies are breaking down the barriers that have kept information from flowing between different brands of computers and software. First there was the Internet, the medium that shattered pre-existing models of deploying Information Technology (IT) solutions. Then there were new application development tools such as Java and ASP, which reflected a new generation in software engineering. Now there are emerging standards such as XML and SOAP, to which some will say is the proverbial “icing on the cake.” Vendors will continue to hype their products, as being capable of creating phenomenal shifts in the software business that are “no less seismic than the fall of the Berlin Wall.” With all these new, rapidly growing industry developments it makes sense for us to attempt to gain further insight into the dynamics of the technology marketplace at large, and to focus on what it all means to our clients in the financial services industry.

## The Model

One model commonly discussed during industry analysis is Porter's model of competitive influences. As we discuss the introduction of technology in the organization, it is worthwhile to briefly consider how Porter's model might be applied to the new software industry. It is very important that we clearly differentiate the development and distribution of Java-based (or Microsoft's .NET Framework) applications by users from the development of new products, for which development costs can be a significant barrier. For this article, our perspective is taken from users within an organization who seek to apply technology to their own business' problems.

Using this perspective, a Porter model analysis might yield the following conclusions:

| | |
|---|---|
| Barriers to entry are relatively low | Notwithstanding budget issues and internal corporate standards constraints, it is relatively easy to introduce new software engineering technology to an organization, |
| Threat of substitutes is high | It is not uncommon to see intense competition to be the "solution provider of choice" between departments, agencies or business groups within organizations, especially high-growth orgranizations. |
| the power of | This element is particularly |

| | |
|---|---|
| developer/suppliers | interesting because depending upon concentration; the power of qualified suppliers can vary. In fact, in many cases, this becomes the single most important aspect in making decisions regarding the introduction of new technology. Emerging technologies are typically characterized by low provider concentration and relatively high power. As the technology matures, this situation reverses. |
| Consumer's wield great buying power | This factor also varies, but to a lesser extent, inversely with the power of developers/suppliers (see above). As the power of developers/suppliers diminishes, the consumer's buying power increases, the mitigating factor being the degree to which the business drivers for the project are strategic. |
| | |

Nonetheless, Porter's model is more useful for industry analysis and does not provide good insight into how and why (or why not) our clients are pursuing object-oriented software development for their purposes. The SPOT Model and the Hull 4 Box Theory[1] are more aptly suited to explaining the

[1] Frank Hull, Fordham University Gradudate School of Business, 1995.

process by which such innovations are introduced to an organization. Before explaining these models further, some higher level observations can aid in forming a better framework for discussion. First, many of our service company clients such as banks, brokerages, clearing houses, accounting firms, and insurance companies rely heavily on computer and software technologies. Second, these technologies are applied just as often to back-office operations (similar to the way in which manufacturing organizations apply technology to production operations) as they are to front-office (i.e. customer service, promotions, etc.) operations, in one way or another. It is imperative that we understand how the use of a new technology is evaluated by an organization

how a new technology is integrated into an organization

how an organization benefits from the use of a new technology

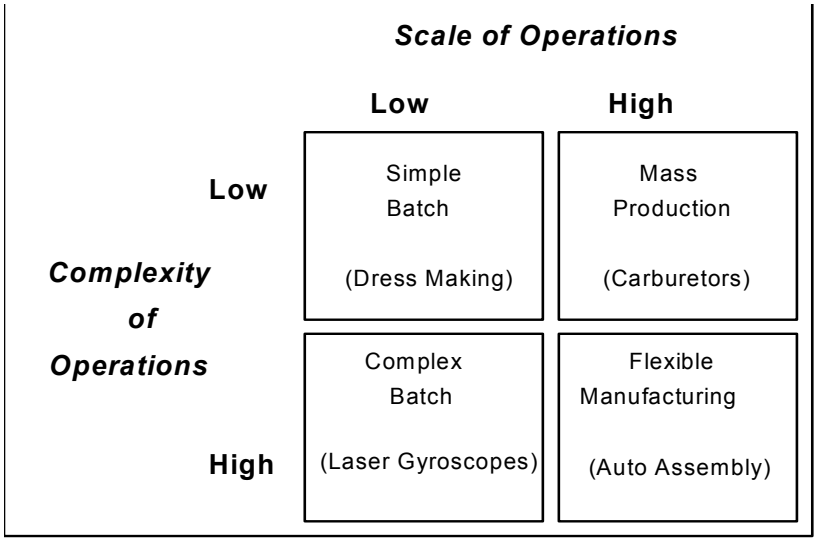


Figure 1 - Characteristics of Process Technology

Several theories exist that model the strategic management of innovation and technology. These models map the methods by which technology is introduced to an organization by examining the level and type of technology employed in their operations. For example, a company with high volume mass production lines will manage technology differently than a company that manufactures laser gyroscopes, or from a company that combines these characteristics and sells large volumes of highly customized products. An organization's production technology can be viewed along two dimensions - complexity of operations and scale of operations. The organization will fit into one of four boxes defined by these dimensions, as shown in Figure 1:

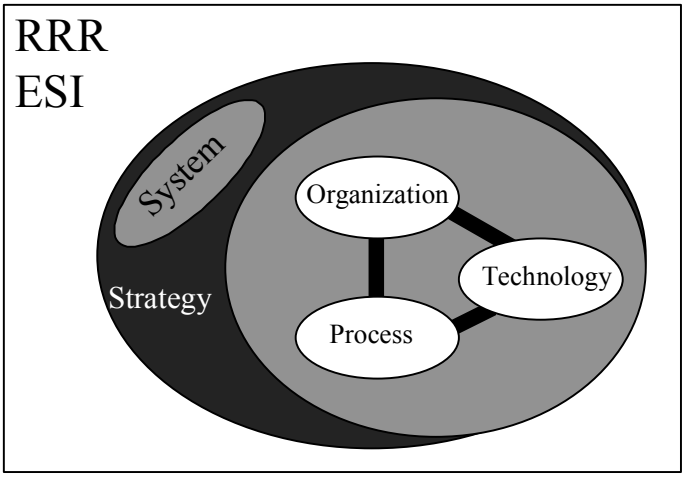


Figure 2 - SPOT Model

The SPOT model addresses the following areas of technology management: Strategy, Process, Organization, and Technology (see Figure 2). The principle focus of the SPOT model is the tradeoff faced by organizations between investments in research and development that result in greater product novelty, and investments in projects that yield

process efficiencies and cost reductions. The SPOT model suggests that organizations can be successful in achieving greater product novelty/inventiveness (as measured by the number of new ideas that make it to market) and/or process efficiencies (as measured by reductions in overhead costs) by applying a combination of techniques such as Rapid Reiterative Redesign (RRR), and Early Simultaneous Involvement (ESI).

The strategy of rapid, reiterative redesign (RRR), proposed as a "best practice" for Concurrent Engineering effectiveness, is equally suitable for service organizations as it is for manufacturing organizations. This concept is little more than the long-familiar rapid prototyping concept that is found in reengineering. In fact, these concepts have been made practical, in part, by the emergence of object-oriented software tools capable of reuse and rapid tailoring. OOP-based development environments such as Visual Basic and Powerbuilder (and now Java) will play an increasingly important role in our clients' efforts to reengineer and retool.

Early Simultaneous Involvement is another technique used to improve development efforts. With ESI, companies expect to maximize the benefits they realize from systems development by involving the early adoptors of technology (i.e. the technologists) with the business, or domain, experts (in service companies, these will be the analysts, underwriters, traders, researchers, and accountants) earlier in the development process (see Figure 2). Early Simultaneous Involvement is only possible if the communication barriers between the technologists and the domain experts are broken down. Using OOP-based development environments such as

Visual Basic, we have had higher rates of success by pushing forward the development of applications in ways that enable users to see and touch a product much earlier in the project life cycle. Imagine how impressed our clients will be when we can combine these capabilities with the ability to more easily distribute greater functionality across wider geographic areas and platforms.

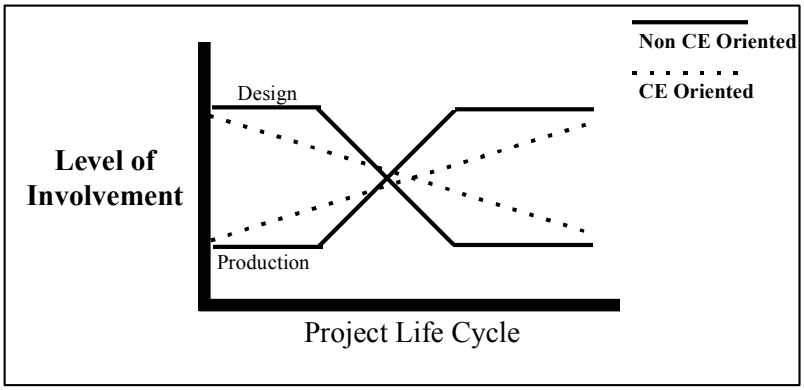


Figure 3 - Early Simultaneous Involvement

Consider the current cost to a corporation for the conversion to a new suite of software, or the cost of developing and deploying a new internal application. Currently, the annual cost of supporting a corporate PC user is $8,000.[2] Corporations are weighing the the value of upgrading software every two years or so. Look at how companies took a go-slow approach to Windows 95, Windows 98, Windows 2000, Windows XP, and .Net because of the huge costs of upgrading. The inability of companies to capitalize on the promises of reusability and ease of use of object software in general has been the biggest stumbling block for software technology. Training costs, as well as development and

[2] Gartner Group, Inc.

deployment times often exceed projected estimates until the developers become accomplished object experts.

With the prospect of software becoming smaller in size and more easily distributable as a direct result of advances in technology such as Java and the Internet, object oriented technology can begin to deliver on its promise to be of true value to the masses. As technology leaders we will are already becoming familiar with the practical application of these concepts. As a relatively immature technology, opportunity abounds.

## OOPS

The commercialization of object-oriented software began in earnest in the 1990's as numerous companies sprouted and attempted to proliferate object technology across various computer hardware platforms. With the emergence of several object-oriented programming (OOP) standards (i.e. Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA)) and object-oriented versions of languages such as MS Visual Basic (and yes, even Cobol), programmers took advantage of the benefits of this new approach. Similar efforts were underway at large software development houses such as Microsoft, IBM, and Apple as well. However, for a long time, the majority of benefits were realized only in the more technical communities, not by information technology (IT) consumers at large. Languages based on objects (i.e. C++), high-end object-based development environments (i.e. Visual C++), and specialized versions of cross-platform object environments (i.e. IntelliCorp's ProKappa) were embraced, for

the most part, by early adopters and "bleeding" edge developers. Object technology remained in the early stages of product maturity. However, the world of software as we know it has dramatically changed, and along with advances such as the Internet, object-oriented software now offers incredible new ways of accessing information and achieving productivity increases for developers and IT consumers alike.

Through such concepts as reuse, encapsulation, and polymorphism,[3] object-oriented technology facilitates the creation of modular software components, reusable objects, and encapsulated applications. Applications designed using this technology can easily be tailored to fit the business needs of a company. Unfortunately, spreadsheets, databases, and word processing software written for one platform still do not work on others, and software running on the same platform still doesn't work well with other programs. Software vendors have responded by applying object oriented programming (OOP) techniques to develop large "suite" application programs that make it appear that these various applications work together seamlessly. The fruits of OOP have become widely visible in most commonly used business packages, for example Microsoft Office, Wordperfect Office Suite and Lotus Notes.

Today, major software products and industry advances are evolving from both OOP *and* the Internet. This is the natural

---

[3] Reuse implies that a segment of code can be used by independent programs without further development. Encapsulation describes the ability of a unit of software to be functionally independent by packaging both code (behavior) and data in the same container. Polymorphism is the characteristic of a software object to be called one name, while applying it to many different contexts.

evolution of the long-recognized trend of faster processors colliding with faster networks. Although software developers have forged ahead and delivered better and wider selections of horizontal and vertically oriented software tools, they have been frustrated by limited deployment options. Concurrently, workstation and telecommunications vendors like Sun, DEC and AT&T have succeeded in establishing truly global network connections (and let's not forget the hard work and assistance of NASA in "installing" the necessary satellite connections!).

## Java and .Net

All these advances have recently culminated in a software product called Java. It is not surprising that Java was developed by Sun Microsystems, the company that has long been promoting "the network *as* the system." Java embodies two key attributes of the new world of software: it is designed specifically for the Net, and it is an OOP-derived tool capable of bringing the benefits of OOP to the masses. One definition of Java that I recently came across is "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language." (Wow!) Java has the potential, in the long term, to change the way in which consumers obtain and use software. For the first time, highly functional software will be on-line and easily accessible to the networked community. Sun's vision for Java is that its compact applets, many taking up less than 100,000 bytes, will do a single job well. If a user wants another feature --say, a spell-checking on a word-processor, or a graphic chart -- they simply click to fetch

another applet, which arrives in few seconds. Java thus offers users the tempting prospect of a virtually infinite supply of just-in-time software, passing the burden of storing it to the network. In effect, using Java, the Internet will become a large virtual disk drive capable of storing just about everything, and by acting like a huge processor, capable of performing any computer-based task on any platform that can be connected. [Note: At this time, I'm reserving judgement on the notion of people throwing away their full-blown Pentium PC's in favor of simple, cheap Internet-boxes. It's not that I don't think that these machines will be available, I just think that people will want to retain substantial processing power locally. Remember when IS mavens thought X-terminals would replace workstations?]