Assessing New Technologies & Products: Insights to Consider

William Barry
Director of Administrative Computing
Dartmouth College
william.f.barry@dartmouth.edu

Abstract
Using new technologies often requires information technology (IT) management to confront complex decisions. Products, methodologies or entire market segments can be transient. Previously unmet needs escalate. Frazzled IT professionals are often swept along like lemmings racing towards the latest answer to our prayers, while vendors and consultants embellish their products, with an exaggerated hype that is sometimes surpassed only by our own IT colleagues.

This paper is an overview of lessons learned from past innovations in technology. While acknowledging the successes of each technology, examples will highlight overblown expectations, unrealistic technology maturation timetables, premature obituaries for existing technologies, and mythologies of cost reductions or productivity gains. Specific technologies referenced will include: expert systems, open systems, CASE tools, object oriented methods, client/server architectures, middleware, fat versus thin clients, web technologies, the role of mainframes and trends in centralized or decentralized organizational structures.

This paper will include guidelines on the need to balance elements of risk in a portfolio of IT projects and considerations to help reduce risk with emerging market technologies.
Investing in new technologies often requires information technology (IT) management to confront a complex, often bewildering, assortment of options. Products, methodologies, vendors and even entire market segments can appear, disappear or be made to look terribly shortsighted within the implementation schedule of even fast-track projects. Past deliverables very often fall short of users’ escalating needs. Frazzled IT professionals sometimes encourage and often are swept along by the stampede of lemming-like behavior racing towards, at last, the latest answer to our prayers. Meanwhile vendors and consultants embellish the true progress of their products, with a level of hype and exaggeration that is sometimes only surpassed by our own IT colleagues.

This paper is intended to provide IT management with an overview of lessons we have (or should have) learned from past innovations in technology. While acknowledging the successes made in each technology reviewed, this paper will highlight examples of overblown expectations, unrealistic technology maturation timetables, premature obituaries for existing technologies, mythologies of cost reductions and productivity gains.

Rising Costs & Dashed Hopes on the Road to the IT Promised Land

In 1983 Warren McFarlan, in describing how companies face repeated cycles of new information technologies, wrote: “While the company’s use of a specific technology evolves over time, a new wave, that is a new technological advance, is always beginning, so the process is continually repeated. ... as the costs of a particular technology drop, overall costs rise because of the new waves of innovation.”

Given the dramatic improvements in the price/performance ratio of computing hardware, as well as the instances where automation can produce substantial labor savings, it seems counter-intuitive that technological advances could be accompanied by an overall rise in costs. With all of the enthusiasm that often accompanies new IT, and the perennial proclamations of cost reductions and gains in programmer productivity, why is it that the total costs of computing seem to continue to rise?

Decisions to adopt an emerging technology should be based upon a realistic assessment of the technology’s maturity, the relative magnitude of resources being committed, considerations of the cost/benefits of using more familiar technologies and best-estimate prognosis of the productive longevity of the new technology. However, more frequently than commonly acknowledged, such decisions are made in a climate of frustration and dissatisfaction with the current state of installed IT solutions. For example, Casper Jones has reported in 1991 that the average MIS project is one year late and 100% overbudget. It is within this climate that IT investment decisions can often be based more upon wishful thinking and the delusions of false promises instead of well-reasoned assessments of cost and risk factors.

The arrival of many new Information Technologies is accompanied by a recurring pattern of hyperbole. Perhaps it is partially due to the extent of our unresolved IT needs and the pressures we face to deliver on IT promises, that the computing industry often repeats a similar pattern with each new technology. This pattern includes an initial period of euphoria and elevated expectations which leads to a peak of exaggerated promise, followed by a period of disillusionment, then pragmatic assessment. At the end of this cycle the new IT often becomes a respected and productive component of the established repertoire of IT tools. Represented graphically in Figure 1, this pattern of exaggerated expectation followed by pragmatism is referred to as the Hype Cycle of new IT.

Perhaps the most difficult period of emerging technology assessment is during the period of euphoria and inflated expectations that accompany the early cycle of emergence. During that period, there develops a wave of enthusiasm that is driven primarily by innovative pioneers and embellished marketing. The pragmatic judgment of lessons of past overblown IT expectations tend to be ignored in the high expectation excitement of what is perceived to be breakthrough technology. The user community, which so often is forced to accept compromised or delayed IT solutions, is quick to grasp at the promise and

---

excitement of a new technology. These factors set the stage for leap-before-you-look tendencies that encourage over optimistic expectations of project deliverables, timetables and costs.

The extent to which this scenario occurs can vary for each emerging information technology. When this scenario of overblown expectations and exaggerated hype is especially severe, it becomes politically correct to not speak out against the rising hype of an emerging technology. In that climate, one risk is that the council of experienced IT specialists, whom are often consumed with the challenges of more mature technologies, is often not given sufficient credibility nor consideration. Experienced IT professionals also can get swayed by the apparent promise and growing momentum of the new technology. In an industry where the IT professional often questions or raises complications involving the user requirements not always apparent to the user, it can be especially hard to speak negatively against the rising tide of inflated expectations during the early hype cycle of a new technology.

**Expert Systems Promises**

During the 1980s Expert Systems and Artificial Intelligence spin-offs were the subject of much acclaim. Promises were made by vendors and proponents that these tools were going to create knowledge-based systems that would capture and leverage the expertise of a human specialist. It was proposed that such expert systems would revolutionize applications ranging from medical diagnostics, to classroom instruction to psychological counseling. These overly ambitious predictions of imminent success were followed by disappointments, and delays or failures due to the complexity of the rules and processes being encoded or the maintenance of rules-based codes that proved just as burdensome as the if-then programming logic of earlier tools. By the late 1980s, in a reaction to the over-hype of the earlier enthusiasts, AI and Expert Systems approaches became an anathema.

---

3 Gartner Group, Advanced Technologies Management Research Note, 1996
By the early 1990s, with less fanfare and hype, many of the more modest goals of expert systems have been realized in systems such as credit approval processing, inventory management and manufacturing control. In writing about what we should have learned from this over-hyped technology, University of Texas Professor Tom Davenport writes, “We should realize by now that the level of hype about a new technology has little correlation with its ultimate success... The more grandiose the predictions about a new technology, the greater our skepticism should be.” This is not to say that the future of Expert Systems technology is without promise. There continues much research and some product development in the Artificial Intelligence arenas of case-based reasoning, fuzzy logic and commonsense modeling... More progress will come but perhaps now the challenges are more kept in perspective.

**Open Systems — Concepts and Promises**

During the late 1980s the concepts and promises of open systems produced much trade press coverage and many advocates. Perhaps inspired by a time which saw the crumbling of such closed or control dominated institutions as the Soviet Union or IBM, many IT professionals and users were enticed by the open systems promise of freedom from proprietary vendors, increased interoperability of software, and the ease of porting systems regardless of hardware size or manufacturer. This was a period of tremendous hype and overblown expectations involving the laudable goals of open systems. In principle, an open system is a system that is able to communicate with or within other hardware or software systems according to sets of formalized communications rules referred to as standards and protocols.

Much effort continues to be made to define open protocols for multi-vendor database interoperability or multi-systems remote program calls. While there were many positive gains resulting from the open systems initiatives, especially in the area of improved interoperability of systems and the evolving open web technologies, the reality has fallen far short of the promise. The IT trade press is frequently reporting on examples where open systems standards have fallen short of expectations. A representative example can be found in a 1994 Datamation article aptly titled “Lies, Damned Lies, and Reviewers” which outlines the failures of interoperability and consistency of several current database and electronic messaging standards. This theme is also well described in the area of Open Database Connectivity (ODBC) standard problems described by Richard Finkelstein in a 1994 Computerworld article titled “ODBC Spells Headache”.

The UNIX operating system was often seen at the vanguard of what it meant to run open systems. The politically correct popular question for IT managers became “when are you going to UNIX?”. What has happened to the promise of UNIX to provide vendor independence from proprietary operating systems and hardware platforms? As I wrote in the Winter 1994 issue of CAUSE/EFFECT, one of the fundamental premises of open systems runs counter to the goals of product innovation and differentiation that drives the free market; “Hoped-for realization of further commoditization of software components will be achieved slowly, since this trend runs against free-market economic forces. In a marketplace where a vendor’s product differentiation determines market share and survival, vendors will continue to resist making their products inter-operate.”

During the past few years, it has become apparent that the UNIX market has begun to further differentiate. According to David Linthicum, “In the hands of vendors trying to compete with each other, UNIX has become a series of proprietary monstrosities that can’t run each other’s application code.” This is not to say that UNIX is not a viable and important operating system. This is only a reminder that there exists no single operating system, proprietary or not, that is a panacea to meet our needs. What has become of the open systems goal to move away from proprietary vendor solutions? A Computerworld 1994 survey found that “... when Information Systems managers were asked which standards must be adhered to when buying information technology, de facto product standards outpointed open systems standards 2-to-1.” In 1996, The question of ‘when are you going to UNIX’ does not seem

---

8 Maryfran Johnson, “It’s Open - Really!”, *Computerworld*, Jan. 4, 1993, p. 8
12 John Foley, “Unix Unification Falls Short”, *Information Week*, June 5, 1995, p. 96
to be as popular, perhaps since this is the first year in which sales of WindowsNT (a proprietary operating system) have exceeded the total sales of all variants of the UNIX operating system.

From the perspective of the business community, open systems are “Great in theory, a quagmire in reality. ‘Standards’ and ‘open systems’ have become perhaps the most overworked and meaningless words in the computer industry, mantras that every supplier feels obligated to chant as often as possible.” 17 If delivery on the promises of open systems are to be late, if realized at all, the failure was due to the computing industry’s lack of successful development of standards. Design based upon standards is a fundamental construct of engineering. Yet the computing industry has, in most cases, failed to achieve sufficient standards.

“Computer standardization turns out to be one of the field’s biggest myths. The industry has done a shockingly poor job developing and adhering to consistent standards that allow computers to work together easily and expand inexpensively.... businesses spend large sums buying products and paying staff to overcome the incompatibilities nonstandard systems create. A lack of standardization raises hardware and software upgrade costs; raises training costs by lengthening the learning curve; reduces the flexibility of the computer resource; and either prevents computer-to-computer communications altogether, or renders it vastly more expensive.” 18

In conclusion, the lofty goals of open systems, which can be easily sold to deans or vice presidents dissatisfied with installed systems, are quite often yet-to-be delivered, or achievable only at substantial costs. Those who have profited most from this talk of open systems may be the silver tongued consultants who promote it or the contract programmers which many of us have hired to make these “open” systems talk to each other.

CASE Tools — Mostly Un-met Promises of Programmer Productivity Gains

Since at least the early 1980s, Computer Aided Software Engineering (CASE) tools have been evolving through several cycles of promises and disappointments 19. While there is debate about how to define CASE, a working definition is that CASE is “a combination of software tools and structured development methodologies” 20. The promise of streamlining or automating the programmer’s work is the essence of the CASE vision. Early tools provided support for discrete portions of systems analysis and development labor. More recent tools attempt to provided integrated facilitation or automation of these programming tasks. Representative of both the vision and the hype of early CASE promises, in 1985, James Martin, a prolific IT author, wrote about mathematically provable correct program code. Martin wrote about the lofty goals that are at the essence of most CASE product hyperbole.

“The technique has been automated so that bug-free systems can be designed by persons with no knowledge of either mathematics or programming. The software automatically generates bug-free code... (this technique) has been applied to highly complex systems. The technique is used not only for program design but, perhaps more important for high level specification of systems. The design is extended all the way from the highest-level statement of system-functions down to the automatic generation of code.” 21

A whole segment of the IT industry has grown around CASE. The persistence of the CASE products marketplace, is sustained by the need to solve what for decades has been referred to as the ‘software crisis’, i.e. the need to improve programmer productivity. Despite gains made in many systems development technologies, this crisis continues to get worse due to the growing complexity of application development environments and user business requirements. Edward Yourdon writes, about using CASE, that “…the risk of failure is ever present: it is quite possible to spend millions of dollars on CASE tools, and then discover a few years later that it had no impact whatsoever on productivity or quality.” 22 Problems with CASE tools & methods include: poor integration of tools, lack of multi-vendor cross-platform compatibility, lack of methodology or standards in an organization’s software development practice, confusion or over-expectation about the role of the CASE tool, and missing functionality in the CASE tool

19 Edward Yourdon, Decline & Fall of the American Programmer (New Jersey: Prentis-Hall, 1992) pp.132-135
20 Carma McClure, “CASE Experience”, Byte, April 1989, p. 235
22 Edward Yourdon, Decline & Fall of the American Programmer (New Jersey: Prentis-Hall, 1992) pp.156-157
being used. Perhaps one of the largest single factors of CASE failure is the lack of understanding that it is intended as a tool to aid Software Engineering, which is the phrase used to describe late 1980s attempts to formalize systems development into an engineering discipline. Highlighting these problems in not intended to portray CASE as being a total failure. Studies show that an increasing number of projects are achieving success with CASE, however, some estimates place the use of CASE tools within US companies as low as 2%.

RAD Tools for Rapid? Application Development

Rapid Application Development (RAD) is promoted as a combination of intensive user-requirements definition sessions, CASE tools, system prototyping techniques and iterative development and testing cycles that are intended to provide greatly accelerated IT implementation timetables. For small systems, local to a single department or highly structured business function, RAD techniques have been of value. The use of RAD tools at Cornell University, as reported by David Koehler, in the 1992 article “Adopting A Rapid Application Development Methodology”, resulted in some modest successes and disappointments. RAD as a solution to the need to more quickly implement large systems projects, is not yet ready for prime-time. For large institution-wide systems projects, the issue was well summarized by Leonard Mignerey,

“Extravagant claims of success notwithstanding, there have been a number of stumbling blocks to achieving ...(RAD’s goals). The business rule component is a major difficulty. The complexity in coding applications that are mission critical to the entire enterprise is orders of magnitude more difficult than developing department-level, non-mission-critical applications. Some ... products have been fairly successful in dealing with the general rule portion; however, it is the exceptions to the rules that are the Gordian Knots of RAD”.

Object Oriented Software Development — Early Promise, Slower Than Expected Adoption

At least as far back as 1981, proponents of object oriented programming were forecasting that the software development industry would be dramatically changed by the methods of object oriented technology. The promise of reusable libraries of objects, with each component designed to accomplish a specific building-block task, was promoted to provide improved programmer productivity and a reduction in system bugs. The pace with which these tools and methods evolved has been very slow. Reasons for the slower adoption of this technology include: the concepts and methods can be resisted by many traditional my-code-is-the-best style programmers; the initial project that uses this technology can take longer than using alternative approaches due to the need to build object libraries; and the different style of thinking about object encapsulation, inheritance or polymorphism of programs can represent a steep or unattainable learning curve for some programmers. Despite the slow progress of these tools and methodologies, there are some indications that the penetration of object oriented approaches in systems development projects is finding gradual but increasing success. However, the common perception of most IT professionals, regarding object oriented approaches, may be represented in the flowing quote, “People say that object-oriented tools will get us out of the current mess, that we will have libraries of objects that we will just assemble to create new applications. Unfortunately, we are a long way from the realization of this vision - if it is a vision, and not a pipe dream. Most organizations are no more ready to have shared libraries of objects today than they were to have shared corporate databases in 1965.”

Client/Server — Promises, Delusions and Experiences

From approximately 1991 through 1994 the IT industry saw the rise and peak of unrealistic hype that accompanied the concepts of client/server as a ‘new’ information systems architecture. The essence of the client/server model was that application software was to be partitioned, with system components located on the optimal hardware platform, often times a database or file server accessed by a desktop client set of programs. Concurrent with trends in corporate downsizing and the then popular myths of open systems, plus the predicted demise of centralized computing’s mainframe-centric solutions; client/server systems were frequently touted as a means to lower costs and decentralize control of systems. During 1991 and 92,

23 Carma McClure, CASE is Software Automation (New Jersey, Prentice Hall, 1988)
the rising crescendo of hyperbole promoted client/server as a paradigm-shift in computing. During that time, it was politically correct to promote client/server as the solution to most all IT needs. Within IT circles the popular question became, ‘when are you moving to client/server systems?’ At that time, it was difficult (i.e. politically incorrect) to question the trend and perceived promise of client/server solutions. This was despite client/server’s lack of: mature software development tools and debug utilities; security controls; operations management utilities; and distributed code-management tools. By 1993, client/server enthusiasts were beginning to acknowledge that client/server systems would often cost more to create and support than mainframe-centric systems, but that the added functionality and the GUI(Graphic User Interface) made possible were rationalized to be more important than the higher systems costs. The trend to ‘downsize’ data processing and business software logic from centralized computers to client systems continued to be encouraged. Towards the end of 1993 and 1994 the client/server hype cycle’s peak of expectations began to crash into the sober realities that IT staff experienced with distributed systems had been familiar with since at least as early as the mid 1970s. The client/server hype hit the reality of the greater complexity of systems development, the immaturity of client/server tools 31,32,33,34,35,36 and the greater labor costs of implementation and support 37,38.

**Middleware — Let’s ‘Black-Box’ the Client/Server Problems**

With the early 1990s rise in client/server enthusiasm, the term “middleware” was created to refer to a mix of innovations which were intended to extend operating systems and programming tools to solve some of the challenges of client/server computing. Proponents of middleware solutions correctly point out that middleware is essential to solve the distributed computing problems of hardware independence, interchangability of system or database components, and the need to mask the complexities of heterogeneous operating system and network protocols from the software developer. It is my contention that middleware is mostly a convenient conceptual black-box that has been used to brush under the rug what client/server approaches have not yet resolved. In the computing trade press, I find support for my contention, as evidenced by the titles of the following articles: in the May 1993 issue of Database Programming & Design, “Middleware or Muddleware? Finding the Right Database Solution” by Colin White; In the April 1996 issue of BYTE, “The Muddle in the Middle, by John R. Rymer; and in the same issue of BYTE Middle(ware) Management - Don’t Panic! Middleware tips from the Experienced can Smooth Over a Project’s Bumps”, by Salvatore Salamone. The middleware challenges are summarized by Edmund X. DeJesus, who writes “… programmers today face impossible tasks: dealing with multiple APIs, making applications portable to any network, connecting to any database. All the ‘standards’ that were supposed to make their lives easier didn’t. Instead, they must know the specifics of each one to create their applications. Nobody has time for this mess.” DeJesus goes on to state that middleware is intended to solve the mess. Whether that is to be accomplished remains to be seen as middleware is still evolving, along with the various systems products and techniques that it is trying to lend coherence to.

**Premature Obituaries for the Mainframe**

From the mid 1980s through the early 1990’s, the many successes of personal computers, local area networks(LANS) and decentralized departmental systems led to many calls for the end of the mainframe. The connotation of the word legacy became mostly negative when used as a label for existing systems. The high costs of centralized IT organizations, during times of fiscal pressures to downsize, became much more noticeable than the less visible, but growing costs of decentralized systems. The old model of astronomical software licensing and hardware maintenance costs, tied to large ‘big iron’ mainframes was only beginning to be replaced by less expensive per-user mainframe software licensing practices and smaller central computers with better price/performance metrics. All of these factors fueled the early hyped promises of open systems. Within this context, the presumed Bastille-day cry of industry pundits and PC-enthusiasts pronounced a revolutionary IT upheaval in process that would mean the end of the mainframe-era. The few experienced IT professionals who attempted to question the inevitability or timing of the mainframe’s death, were seen as the close-minded old-school mentality, most of whom were to be denigrated or pitied as the last dinosaurs of a dying era. Like most generalized stereotypes, this

37 Jeff Moad, “Client/Server Costs: Don’t Get Taken for a Ride”, *Datamation*, Feb 15, 1994, p. 34.
perception was certainly accurate in some cases. The frequent occurrences of exciting breakthrough IT solutions involving PCs or decentralized systems reinforced the lack of confidence in the old mainframe department and its staff who were too often bogged down in maintenance support of existing systems to keep up new technologies or practices.

The accelerating hype of the death of the mainframe, lead many organizations to issue an executive mandate: “no more mainframe systems development”. IT professional associations began to conduct surveys asking about “plans to eliminate mainframes”. Many universities appointed computing planning task forces, often led by deans or faculty members with no understanding of the business needs or technical challenges of large institution-wide systems or the still unsolved technical challenges of distributed computing. On many campuses, this lead to reorganizations, costly consulting engagements and frustration-initiated decisions to reallocate budgets in favor of new technologies or systems-replacement projects. With such new initiatives in their early stages, while the new IT solutions were still on the unrealistic expectations side of the hype cycle, many organizations proceeded with a newfound sense of optimism.

As the euphoria of what was expected to be a new computing paradigm began to fade into the all-too-familiar IT project delays due to immature products, unstable standards, and underestimates of the complexity of user requirements; the frustrations often associated with mainframe legacy systems projects started to appear within these new IT ventures. The problems of cost underestimates and missed timetables occurred with increasing frequency. In 1996, systems development in a non-mainframe environment still suffers from a lack of adequate tools for efficient: systems management, operations scheduling, programming debug utilities, version control needed to coordinate distributed systems components, and security.

In the business and IT trade press, from about 1991 through the present, there was a gradual increase in reports that the costs of distributed client/server systems were higher than the per-user costs of systems running on a mainframe. According to a 1993 Wall Street Journal report, “Indeed, boardroom disillusionment about the pace of downsizing has prompted some analysts to think ... that the demand for mainframe computers could surge as the companies realize that the downhill shuffle isn’t all it was cracked up to be,... the downsizing hoopla has promised too much too soon and that companies ought to scale back their expectations”. According to a 1994 study by the Los Altos California based International Technology Group(ITG), in their “Cost of Computing” report, the per user five-year total cost of ownership of a PC-LAN based system is $6,445 per year compared to $2,282 per year of a mainframe-based system. Nearly identical per user cost comparisons are found in a 1996 ITG study, which also reports that the average cost per transaction is $0.03 on a mainframe versus $0.46 on a PC-LAN system. Similar findings have been concluded by several Gartner Group studies.

Although many client/server projects ran into problems, enough client/server systems succeeded to create new demands for large central servers, i.e. mainframe-class machines. This boosted sales of mainframes. The improved economies of CMOS chips and parallel processing or improved manufacturing of mainframe-class machines, resulted in improved price/performance economies for large-scale computer center operations. Some products that early advocates saw as the antithesis of the mainframe model, such as UNIX or Web Servers or Object-Oriented tools, began to fulfill valued roles on mainframes.

By late 1996, it has become clear that the computing resources of the foreseeable next ten years will include centralized mainframe-class machines as one of the platforms on which IT services university and corporate computing needs.
Centralized Versus Decentralized Models of Computing

Discussions of the pros or cons of mainframes often involves arguments about the merits of centralized versus decentralized computing. Perhaps as an outcome of dissatisfaction due to unmet business IT needs, the trends of the late 1980s were towards an increased decentralization of IT resources. This trend was encouraged by the opportunities, success stories, and expectations of several information technologies highlighted in this paper. Since the early 1990s, as the higher total costs of most distributed computing systems began to be understood, there are now signs of an increased understanding of the merits of centralized IT management. An excellent discussion of these issues was written by University of South Carolina Professor of Computer Science, Martin Solomon, titled “The Need to Rethink Decentralized Computing in Higher Education”\(^{51}\). Among other references on this issue, Paul Strassmann, the author of The Business Value of Computers (1990) and the Politics of Information Management (1994) states that centralization of IT projects is more often associated with success and productivity than experienced with a decentralized approach\(^{52}\). The wisest approach is to avoid a view that is tied to only one hardware platform to the exclusion of all others. The many successes and failures of decentralized computing, as well as those of centralized IT, should have taught us that the future of IT includes the full spectrum of hardware, from the mainframe to departmental servers to desktops, all linked by increasingly faster capacity networks. There is no single right answer that always points to only one class of computing hardware. The weight given to the associated IT deployment decision factors will vary per IT resource, from a small college, to a multi-campus university. Unfortunately, we’ve perhaps too often seen that ‘re-organization’ is given as an answer within a climate of general dissatisfaction with computing services. It would be wise for university presidents or deans to avoid the following type of misinformed decision making about the organization of IT resources: ‘if the general climate of user satisfaction is bad, then where you’re not (organizationally) must be better than where you are now!’

Web Technologies — Stunning Successes and a New Level of Hype

Since 1994, the rapid growth in the ‘world wide web’ has dramatically triggered an exciting new era in human communications. This success has been based primarily on the effectiveness and relative simplicity of the four initial standards that made up web technologies: Uniform Resource Locators (URLs), the Hypertext Transfer Protocol (HTTP), the Hypertext Markup Language (HTML) and the Common Gateway Interface (CGI) standard for writing programs to interact with HTTP servers. A desktop ‘browser’ program is used to translate objects (i.e. text, graphics, video or audio) organized or retrieved from anywhere on the internet via HTML and an HTTP server into a graphical user interface (GUI) presented on a computer monitor. The ease with which a non-programmer could quickly and inexpensively accomplish attractive electronic publishing of text and graphics was the primary cause of the initial excitement about the web. The breakthrough nature of these web technologies became apparent with the monthly addition of hundreds of thousands of new users of the internet for e-mail and web ‘home pages’. The capabilities of the web democratized and stimulated tremendous personal and commercial interest in the internet which had previously been used mostly by the international academic community. The search for commercial profits from the internet stimulated a burst of new investment in the web. This level of commercial interest in web technologies and the competitive drive to achieve early advantages in this new IT arena has accelerated the pace of web technology innovation while simultaneously elevating the expectations of this new IT. Even with the computing industry’s history of exaggerated promises, the speed and elevation of overblown web IT expectations have taken the IT hype cycle to a new level.

The benefits of web technologies were quickly apparent. The same web page could be viewed from a variety of desktop systems including PCs running Windows, the Macintosh, or a UNIX workstation. The quality of the user interface and the variety of materials presented were better than what was affordable with previous computing technologies. The growing problems of managing distributed ‘client’ software that was confounding many client/server system projects was solved by the initial uniformity of desktop browser programs. The ‘fat client’ problem, i.e. the near insatiable requirements for increased desktop hardware capacity inherent in earlier client/server systems, was solved via the ‘thin client’ footprint of the initially small browser programs.


The success of web-based electronic publishing led to an expansion of HTML to provide rudimentary ‘forms’ tools to facilitate the display or collection of data. Programs created according to CGI or various application program interface (API) standards enabled the integration of databases with web pages that were attractive and easy to use. This led to a much higher quality of user interface for information systems using web technologies that had earned a reputation for being easy, quick and inexpensive. The competitive drive pushing many companies to have a presence on the web required programmers to force the limits of CGI or API capabilities that were incomplete and still rapidly evolving. This lead to many attractive and productive commercial web sites that were often built upon a complex mix of multi-thousand line spaghetti-code programs which used programming languages that are changing monthly, or ‘standards’ for which version 1.0 protocols were months away from being agreed upon. From the user’s perspective, the growing quality and sophistication of these web-based systems further encouraged the rising expectations of the web as a quick and easy solution to most information systems needs. For the support programmer, a new generation of legacy system maintenance burdens was, in many cases, being initiated.

One of the strengths of the initial HTTP standard was its simplicity which allowed for fast transfer of data objects or requests between internet resource providers and requesters. This simplicity was also a weakness that limited the ability of early web technologies to support complex information systems requirements. This led to strategies to extend the capabilities of HTTP servers via either extensions to CGI standards, a proliferation of API standards (some open, some proprietary) and a new class of web tools represented by Java or ActiveX. Concurrent with this was a demand to extend the capabilities of browsers to accommodate a more diverse set of data objects such as video, audio, Java programs and various proprietary-data storage formats. Within the fiercely competitive web market, this has led to product or technology ‘standard’ differentiation that has already begun to compromise several aspects of the initial simplicity of web technologies. ‘Thin’ browser ‘client’ programs are rapidly getting fatter, and thus requiring more desktop machine resources, with each new browser release or additional proprietary ‘plug-in’ that needs to be added to the basic browser to enable access to a growing richness of web page data being presented. Despite the auto-configuration of many browser plug-ins, the increased diversification of browser components has increased the level of technical sophistication required of the user. The variation of browser capability and a users local network access bandwidth combined with the growing richness of data included in web pages has caused an increase in the number of commercial web sites that have applied the increased labor necessary to program two or more parallel sets of HTML and CGI code, to allow for access via high versus low bandwidth users or to allow those with or without Java-enabled browser access. All of these issues are unfolding within a technology that is still evolving so rapidly that standards may be remain transient or elusive and the future market shakeout of tools providers adds to the risks involved. The pace of change and the instability of this technology will almost definitely result in many cases of rapid obsolescence where systems built with these tools may have a productive working life that is shorter than many of us want to anticipate or cost justify. It remains too early to tell the extent to which any of the above concerns may dampen some of the high expectations of web technologies. However, it is clear the web has caused, and is continuing to trigger, dramatic changes in the ways IT professionals need to think about systems architectures.

For large or complex information systems, strategies for web-based systems success should include: thorough specifications prior to development, realistic expectations, and an in-depth technical knowledge of the web tools to be used. As with any IT project, it remains wise to consider all available tools appropriate to a task, web-based or not, and thus don’t choose tools or a design based upon political pressure or industry hype.

JAVA - A New Paradigm?
Amidst the many successes of web technologies, the promise of Java represents an exciting new model of computing. The proven capabilities of Java ‘applet’ programs to provide automatically delivered-when-needed software that can be run without change on a variety of operating systems has begun to break-down the IT professional’s previous conceptions of information systems architecture. The goals of the Java model are lofty, i.e. one set of program code that is portable for automatic distribution to be run without alteration on most modern operating systems. In fact Java, and competitor products like ActiveX represent substantial progress towards the goals of a robust object-oriented programming environment. However, Java is still essentially a procedural language, with mostly character-based development tools, a lack of stable standards, debug utilities or stable object libraries. Java’s ability to access databases is still very much being defined. Version 1.0 of the Java Database Connectivity (JDBC) standard was only resolved
this past summer. However, in the rapidly moving web technologies market, there is a lot of momentum behind Java, ActiveX and the related tools and methods. This will result in substantial improvements in tools, compilers, debuggers, language extensions and object libraries. The degree to which an organization jumps in, or moves slowly should depend upon: the immediate need for Java capabilities not found in more mature technologies, the tolerance level for immature and rapidly changing tools and standards, and the affordability of investing in what may be systems solutions that have a relatively rapid obsolescence due to Java’s immaturity relative to future changes impacting compatibility of the language, standards or operating systems.

**Thin Clients and less than $500 Network Computers**

Has the pendulum swung full cycle? The IT industry has matured from the ‘eliminate all mainframes’ period, through the client/server goal of placing computing processing on the ‘fat’ desktop client, back to ‘thin’ clients that are the new generation of ‘network computer’ dumb terminal connected to large servers that represent the new generation of mainframes. In retrospect this can be seen as an evolution that has resulted in a growing optimization of a broad spectrum of computing technologies. Perhaps a recognition of this full-circle evolution will help put into perspective the next hardware platform-specific solution or paradigm-shift that is touted as the answer to all of our computing needs.

**Guidelines for Assessing New Technologies and Products**

The degree to which the hype cycle of new information technologies has been a factor can vary in terms of the extent to which facts are obfuscated by emotion or wishful thinking as well as the rate at which the cycle unfolds. Despite the hype aspects which have occurred in the IT market segments referred to in this paper, many success stories with each of these technologies could be pointed to as an argument against the hype cycle premises of this paper. Such arguments, when of value to an organization’s current or future IT investment decisions should be pursued. However, a general caution to keep in mind in such technology decisions, is summarized, by the popular IT author, John C. Dvorak, who writes, in an article titled “False Promises”, “The computer industry continues to promise far more than it delivers. And with each new generation of naive user, the promises get repeated like a mantra. A mantra for suckers.” Therefore, don’t be quick to swallow what a vendor or consultant knows you are starving to hear. During product demonstrations or while listening to claims from your staff or co-worker, ask for definitions of such terms as” ‘cheap”, “easy”, “quick”, “open”, or “rapid”.

Learn more about the financial myths of IT. John L. Oberlin’s paper “The Financial Mythology of Information Technology: Developing a New Game Plan”\(^{57}\), provides a thorough examination, from a fiscal perspective, of the following IT Financial Myths:

- Falling computer prices and commodity markets will reduce the total cost of campus expenditures on IT.
- Cheap PCs with the power of mainframes are making distributed computing cheaper than central computing.
- The marginal costs of supporting another software package, hardware platform, or standard is small.
- Information Technology investments can be effectively managed through an ad hoc funding process.
- Personal computers and distributed computing environments mean an end to central computing authority and enterprise-wide standards.
- Emerging Technologies and technology-based services will be cash cows for higher education institutions.

The traditional practices of creating a Request For Proposals (RFP) or performing a thorough function point analysis continue to serve as valuable tools in the assessment of a new technology or a new IT product. Especially for larger projects (as measured by budget or scope of institutional involvement) these methods are essential. The over-hyped promise of a new technology can make it very tempting to assume that a time-consuming careful analysis is not needed. But such decisions should be made relative to the context of how much staff time or capital budget is at risk. An excellent source of sample RFPs is the CAUSE exchange library. A valuable resource in RFP creation or IT systems function point analysis is the 1996 publication, “Campus Financial System for the Future”, published by NACUBO and CAUSE.

What is your organization’s track record of success in the introduction of brand new technologies or mature technologies that are new to your campus? What lessons from your institution’s past commitment or lack thereof should be considered prior to venturing into new projects. Assess your university’s climate for change, For example, Rutgers University Director of Administrative Computing Services, Leonard J.\(^{56}\)

---

Mignerey’s list of Business Issues to be resolved in conjunction with choosing a technology solution. Mignerey’s list reminds us of several pragmatic IT project implementation issues such as: the alignment of the project with current institutional priorities, the communications aspects of creating organizational change, and the costs or risk factors involved.

A 1981 Harvard Business Review article by F. Warren McFarlan titled “Portfolio Approach to Information Systems” contains a delineation of the need to balance the elements of risk in an organization’s portfolio of IT projects. This article offers an essential consideration of the success or failure factors involved in an organizations approach to new IT. Despite the common sense nature of the recommendations that MacFarlan made fifteen years ago, it is surprising the degree to which organizations or IT management can ignore such practical advice when enamored with the exaggerated promise of the new IT products or methods. MacFarlan’s paper discusses an approach to risk assessment in your organization’s ability to tackle an IT project, in terms of the IT chosen and your organization’s placement on scales of:

- size of the initial project (single department versus institution-wide)
- size of the budget and staff commitment relative to entire IT resources
- staff experience level with chosen technology (novice versus expert)
- complexity of application’s data requirements (highly structured versus unstructured)
- commitment of a senior officer to champion the project

To MacFarlan’s criteria, I add the following three factors:

- multiple vendor products involved (few versus many)
- maturity and stability of relevant standards
- apparent pace with which a new technology is evolving

An objective consideration of these factors, when assessing the risks of making use of new information technologies, can help cut through the wishful thinking and reduce the probabilities of project cost overruns or failure.

Is the technology being assessed a part of an emerging market where the inevitable shake-out of weaker vendors has yet to begin? In this case the risks of determining the market survivor can be obfuscated due to the non-existent track record of technology or the vendor. Prior to a substantial purchase within these market conditions, a careful assessment of each vendor’s balance of strengths in the following three dimensions is essential:

- the vendor’s implementation of the new technology relative to the competition
- the vendor’s financial strength
- the vendor’s marketing and support channel strengths

A product that has a strong technological strength relative to the competition, can have a greater probability of surviving a market shake-out and consolidation. In such a changing market, even a modest degree of weakness in the vendor’s financial and channel strengths, can be safely tolerated if the core technology implementation is solid enough as a corporate asset that could survive a buyout or merger. In the converse situation, a vendor’s financial health or strengths in marketing or support channels should not be allowed to create the appearance of a safe investment in a new technology. At Dartmouth College, we recently used these considerations to assist us in the purchase of a product from a lessor known vendor (Planning Sciences, Inc.) which, in our assessment had a stronger multi-dimensional database(MDD) technology than their competition which held larger market share in the emerging on-line analytical processing (OLAP) MDD market. In this case, at Dartmouth College, the safe option of making a small initial purchase under such market conditions was not chosen. We decided that, in contrast with the somewhat conservative nature of the rest of our portfolio of IT projects, we were willing to take an increased risk, in an attempt to better leverage the investments of the project’s staff time commitments and the opportunities of more aggressively priced site-license options possible with a vendor who was trying to break into the higher education market.

In conclusion, beware of single-answer-to-all-need solutions, whether they are articulated by your staff, a consultant or a vendor. While further progress on many IT technology fronts will be made, lessons from the past tell us that single-answer solutions should always be viewed with caution and almost always are.

---

suspect. We can recall times when local experts or trade-press pundits gave a quick answer, such as Client/Server or PCs before they knew what the question was. Is “The Web” the latest instance of this naive theme? Biases towards any one technology or hardware/software platform to the exclusion of others should be evaluated carefully. If it sounds too good to be true, it probably isn’t true!

In a statement attributed to philosopher and writer George Santayana, “Those who cannot remember the past, are condemned to repeat it”.