Infrastructure Agility

Agility – the ability to move with a quick and easy grace – has always been a quality prized by athletes. But by businesses? Most organizations can rush a particular project or two into completion. However, rapid movement in almost any industry is no longer the exception, but the norm.

Businesses are adapting new processes and models with increasing regularity. The nature and frequency of these changes are creating discontinuity in existing business processes and models – the likes of which have not been seen since the first stirrings of the Industrial Revolution. Information sources, products, and markets that were once highly specialized have been introduced to new players, with highly unpredictable results.

Such discontinuity is also stressing the information systems upon which these changing systems and processes rely. Information technology (IT) organizations are being asked to develop new, mission-critical systems for unknown, untried business models, often using immature or untried technology. Not surprisingly, few companies have developed the underpinnings – the infrastructure – necessary to handle change in a graceful way.

We call the ability to handle change gracefully *infrastructure agility*, and we believe it is a quality that has become increasingly important for business survival and success. IT must develop an organizational infrastructure that enables agile business responses to ephemeral opportunities. Unfortunately, most current infrastructure development projects are rush jobs. Although a quickly developed e-business infrastructure may keep a company stable for six months to a year, it jeopardizes the capability for sustained change, putting the company at risk.

Building an agile infrastructure revolves around three key directives:

- Architect for agility
- Leverage agile technologies
- Organize around critical skills

Architect for Agility

Architecture is the fundamental component in creating an agile infrastructure. It comprises a set of technology-independent rules developed jointly by business and IT. *Architecting for agility*, the focus of this white paper, involves the assembly of various core technologies that make up e-business infrastructure. An agile infrastructure enables IT staff to plug and unplug these technologies as circumstance and requirements dictate. Creating this architectural foundation will also enable businesses to *leverage agile technologies*, which can be quickly and effectively introduced into such an architecture; and to *organize around critical skills*, where businesses minimize investment in scare skills while maximizing the ability to change.

The essence of infrastructure agility is the degree to which developers can foresee which parts of the infrastructure need the ability to change independently. This is accomplished through the following:

Partition Functionality

Partitioning the infrastructure into discrete parts – firewall, Web server, proxy server, domain name server, etc. – involves isolating the parts from dependencies on components within the system. This enables IT to replace a specific technology without disrupting the rest of the infrastructure.

This concept becomes critical given the growing number of components that populate the typical e-business environment – META Group has identified more than 80 key components that can be critical in such an environment. Even if the 80 components are reduced to 18 broader categories (see Figure 1), it still represents something quite different than what is found in traditional IT architectures.

Figure 1: E-Business Components

While some of these components are approaching "appliance-level" usability – plug it in and it works – there is no question that managing 20, 40, or 80 components is a more difficult proposition than managing the low number of components typically found in a traditional client/server architecture. Still, increased componentization is a must. Buying an all-in-one Web solution, with a Web server, application server, proxy server, database server, etc., is not recommended in this situation, as the individual components are still evolving.

This partitioning of components will differ depending on the business. For instance, in most cases e-mail servers are mature technologies that will not change rapidly. But a brokerage firm may need to review outgoing e-mails from its brokers to clients, checking for inappropriate forward-looking statements or other commentary. Such a firm may wish to add a function to its e-mail server that will enable it to scan the contents of outgoing e-mail, and thus require changes to the server as well.

Partitioning functionality also involves decoupling presentation logic and data logic, while creating certain shared services (e.g., security, directory). Many organizations are familiar with developing separate presentation servers to support various modes of interaction, such as voice, cell phone, or Web browser. Each of these presentation servers utilize common application servers based on common business logic, creating a single, unified view of the customer, regardless of media. First generation e-business infrastructure, however, often focused exclusively on the Web page as a single point of interaction, with little or no linkage to existing assets.

For the foreseeable future, businesses will have to support multiple points of integration and multiple points of interaction. We believe certain technological developments that have been touted as panaceas will continue either to fail to meet implementation objectives or be compromised by business realities. For instance, some vendors and users believe that XML will enable them to present information to any display device – browser, cell phone, interactive voice response, etc. – and combine presentation and application logic. This is not an adequate approach because different devices have different navigational needs, therefore requiring different presentation designs. Navigating through information on a cell phone or PDA is distinctly different from navigating through the same information on a PC.

On the back end, even leading companies often believe they can create a single, universal database that contains all customer, product or similar very large data set information. Due to the speed of business, however, this proposition is nearly impossible to accomplish. By the time IT builds a centralized database, the company will probably have acquired new lines of business, each with their own database architecture, or else divested lines of business. This would require adding new databases or extracting information from the central database. Instead, organizations must support multiple database connections to centralized enterprise services, just as they combine multiple points of interaction in those services.

However, database logic should be decoupled and centralized, so that the distributed databases all present the same structure and interfaces to users. This requires changing business logic design to remove dependencies on database access. XML will play a large role in back-end integration, but it will not solve all integration issues between databases and services. Still, directory, security, and other services should be decoupled and shared rather than being built into each application. IT should push vendors to support both external directory services via the LDAP interface and third-party security systems based on public-key encryption.

Centralize State – Distribute Content

The HTTP protocol is by nature stateless. Each request a Web browser makes to a server represents a new session. Leading e-business organizations are building their Web services using what we call a “stateless farm” infrastructure that can scale to support millions of simultaneous

browser accesses, delivering content that can range from static information to fairly complex transactional data. This design uses a farm of Web page servers, with the same content replicated on each page server. Network redirection technology located at or just behind the firewall distributes the Web sessions among these servers. Therefore, a user might not even access the same page server twice when making two requests on the same page – for instance when adding purchases to an electronic shopping cart while browsing a shopping site. Instead, the more sophisticated sites can direct individual users to whatever page server has the least load on it at that moment.

Higher-level protocols use state management to create the appearance of a continuous session to users by capturing complete data on each hit in a central database and using that data to construct individualized pages as needed – when updating the contents of an online shopping cart after a new item has been selected.

This infrastructure design enables a company to launch a Web site with a comparatively small hardware and software investment and scale up quickly by adding more page servers if traffic starts to grow. Since few companies can predict the amount of traffic they will eventually attract when they launch their site, this enables them to avoid buying more than they need. It can handle sudden, unpredictable changes in workloads. It has built-in fault tolerance to provide uninterrupted service if some of the page servers fail. It also allows companies to phase hardware and software upgrades by testing an upgrade on a portion of their servers and quickly removing it if problems develop.

Regardless of the technology, while business logic can be split across many application and page servers, shared state data must remain centralized. Attempts to use distributed databases to share highly volatile information seamlessly across the enterprise have consistently failed. The best practice is to push volatile state data back from the presentation and application layers into the database but move static content as close as possible to the user. Some companies, for instance, use technology to cache graphic Web page sections at the ISP level, decreasing download time. This has an impact on the higher levels of middleware that enable application services and on server hardware, operating system, and storage requirements.

Relational database systems are the most common choice for powering the persistence engine that maintains shared state data, although object-relational technology is appearing as well. A trend is emerging toward using in-memory databases to complement disk-based databases at the application server level as a method of cutting access time to state information.

Externalize Process

Once anathema to many organizations, turning the management of business processes over to third parties has become increasingly common, Simply stated, it is often more efficient to have a

business partner handle part of what once was an internal business process. For example, some companies subcontract the management of their shipping function to Federal Express, UPS, DHL or another carrier. Airlines are discussing sharing ticketing processes, so that passengers can do ticketing for one airline at the counter of another, spreading crowds across more ticketing positions.

Externalizing processes demands a new approach to building applications. Instead of creating unified, vertical systems, IT needs to document each step in the business process it wants to support, then build a series of modules, each supporting one step, linked together by interfaces. Modules that may be candidates for externalization should use external interfaces, based on standards – the unified modeling language (UML) is one – that can easily be connected to the systems of business partners.

IT should avoid having to retrofit external interfaces whenever possible. Companies that did not have the foresight to build these interfaces have found retrofitting legacy systems with interfaces for the Web – and in the process rewriting those legacy systems – expensive and time consuming.

However, IT cannot use external interfaces everywhere – the resulting system would be too inefficient. Therefore, it must identify potential externalization candidates during application design. Unfortunately, the questions of when, where, and what processes to externalize have no easy answers – they depend on the individual organization's business processes. IT must discuss potential externalization candidates with business users. Designers can also get ideas for what interfaces to externalize and how those interfaces should look by studying the innovations enterprise application integration vendors are introducing to enable process automation across applications.

Leverage Agile Technology

Once IT has designed its agile infrastructure, it needs to leverage agile technology to build it. Creating this leverage involves a few fundamental concepts.

Exploit Packages

The fastest way to implement technology to support a discrete business requirement is to buy a package. However, this is not always easy – the packages that provide the business functionality companies want sometimes come with their own infrastructure that does not integrate well with the rest of corporate IT. The package may also require proprietary programming skills, requiring that staff be trained in skills with limited application or longevity.

Companies should push software vendors – particularly during contract negotiations – to use industry standard application programming interfaces (APIs) to facilitate integration. If the

product with the best functionality also presents major integration problems, consider alternative applications that offer better integration, even if it means losing some functionality.

Once the package is in place, minimize corporate investment in it by minimizing customization. Companies select packages because they provide useful functionality out of the box. When IT has to struggle with the package to get needed functionality, it should replace that technology. Companies that spent years customizing enterprise resource planning (ERP) packages learned that lesson the hard way. When possible choose a package with components that offer maximum flexibility in redesigning its internal logic.

Use consultants or the vendor to install, configure and customize packages. Customer management, ERP, supply chain management, Web enablement, and similar software and the standards on which they are built are evolving too quickly to allow internal staff to economically develop and maintain the skills they need to manage these one-off projects.

Packages represent islands of design. Always maintain a way off that island by keeping as much business logic as possible on a shared application server with a more generalized application and using standard APIs to integrate those packages to the IT infrastructure. Vendors are responding to pressure from users to provide APIs that allow their applications to use business logic kept in external servers. Users should take advantage of those APIs.

Combine Application and Integration Servers

Many application servers are excellent at implementing new business logic. They focus primarily on execution services – e.g., the ability to create new components to meet user requests, thread management, load balancing and failover – transaction integrity and security, and providing only rudimentary "gateway" integration servers.

These gateway services are only adequate to bridge to an integration server and should not be used as a platform for building integration logic on the application server. The integration server's fundamental role is integrating applications into existing business logic. They focus on core integration services, including information transformation and routing and end-to-end process control across multiple applications, rather than building business logic. Users should choose only one application server and one integration server.

The gateway and integration product sets are obviously complementary and the application server vendors are either building their own integration servers (e.g., IBM MQSeries) or are forming partnerships with integration server vendors.

Application servers must be seen as distinct from graphical user interface design systems. Creating new business logic should not be seen as simply creating links to a user interface, a

mistake that is sometimes easy to make when using rapid application development (RAD) systems. Business logic should be a part of the end-to-end workflow, and when it is designed in this way it becomes easier to swap a front-end or back-end component.

Application and integration servers sit on top of interaction services, which track state data through a multi-application business process. Under this are interconnection services, which are store-forward systems with message queuing.

Serialize with XML

Vendor-independent standards will become important at this level as companies automate processes that cross business boundaries to incorporate business partners. Virtually all the inter-enterprise integration vendors have standardized on XML as the format for exchanging information across business boundaries. XML will become as ubiquitous as ASCII and SQL, solving just as many problems. Therefore, companies should replace homegrown formats, particularly those not based on relational data schema, with XML tools, which provide greater agility for modifying and externalizing parts of the business process. However, XML technology beyond the document object model (DOM) should be hidden from developers, who should not be concerned about how information is serialized. If programmers go down to the XML tree level, they will develop dependencies. In a few years this skill will become unnecessary as XML is subsumed into higher level tools, much as HTML disappeared into higher-level Web design tools.

Other external standards, such as the Simple Object Access Protocol (SOAP), are emerging; details about these standards can be found on http://www.w3.org/tr/soap and similar Web sites. Before creating their own XML pages, IT should check these resources for existing proto-standards to avoid reinventing conflicting solutions to the same problem. If no XML document standard has emerged for their application, they should look to EDI standards for guidance in designing an appropriate tree structure. Pioneers in XML tree development should contribute their trees to the standards repositories, where they may become adopted by other users, giving the developer a time-to-market advantage.

XML is a three-tier design system that separates the presentation layer from its core structure. The appearance of data in HTML presentations may be radically different from the representation of that data at the tree level. The drawback of this approach is that working out the tree design independent of the appearance requires extra time. However, companies that instead use presentation logic-based, first-generation technology to get their Web applications finished quickly may get to market faster but will be left behind, as competitors using second-generation XML technology attain greater agility to respond faster to market and technology changes.

Organize Around Scarce Skills

The lack of needed skills is the number-one impediment to change. META Group customers constantly say that they cannot train their existing technicians in new technologies like XML fast enough, and they cannot hire staff with skills in key new technologies. Skills are an organization's most expensive and least agile resources. To solve these skills-related problems, you should:

- Maximize employees' ability to change;
- Minimize investments in rare or transient skills; and
- Partition skills to encourage specialization, thus avoiding generalist experts.

To maximize the ability of employees to change, first give them time both to think about innovative ways to change fundamental business processes and to learn new skills. If the IT staff is constantly dealing with crises they have no time to change. One advantage of the new dot.com companies is they have no legacy, so they can focus totally on innovation. Often the IT staff is busy as a result of the downsizing initiatives of the 1980s and 1990s that left one person doing the job of two, but allowing no free time away from immediate demands to think strategically. Just as IT systems need scalability, so IT staff needs extra capability.

IT needs to staff adequately to allow time for training both in specific skills and new business models. This is not easy because IT staff is in short supply. However, companies can hire people with skills that are no longer in demand – mainframe and COBOL skills for example – and retrain them.

However, IT must be selective in its training and avoid teaching staff members transient or rare skills. To avoid that, do not buy niche products that require training in esoteric skills, even if they offer a short-term advantage – Forte, for instance, is a powerful technology that required too high an investment in training. Many META Group clients used it for one project and dropped it because they could not find people with the skills it demanded.

Similarly, avoid mainstream products in niche combinations – e.g., COM on Tandem. Such rarely used combinations require too many special skills. Recognize and minimize training investments in technologies that are likely to be subsumed into higher-level tools in the near future. For example, many companies invested heavily in training HTML experts only to see their need for those skills disappear as HTML was subsumed into Web page design tools. Similarly, XML will be subsumed into higher level design tools.

Instead of training staff in these rare or transient skills, IT should use consultants and third parties that can be employed only to the extent that those skills are needed. When a company does need to develop internal expertise in these areas, it should train a small team rather than all developers

on staff. Developers who know everything are not only rare but also spend half their time in training.

APIs and the skills they require should be petitioned into infra-, intra-, and inter-APIs, according to their purpose. Intra-APIs are used by program modules to communicate with each other. Normal developers need to be expert in them.

Infrastructure or infra-APIs connect software with underlying infrastructure resources – e.g., software application servers and integration servers. Business logic developers do not need to know the details of these APIs, yet many IT organizations continue to train their developers in this technology. Instead, IT should maintain a small group of infra-API systems developers.

Inter-APIs empower externalization of business functions. Deciding when and where to use external interfaces and what interface styles and semantics to use are specialized skills that are vital to the enterprise, requiring enterprise level thinking based on a full understanding of business principles and knowledge of IT integration techniques and tools such as UML. Normal and infra-API developers are not usually the best choices. Good interface design is the key to agility – allowing proper partitioning for process externalization. Bad design wastes staff training time and forces them to do development work for which they are ill suited.

Too often developers must select the operating system and packaged infrastructure – such as COM, CORBA, Java, etc. – then build, tune and debug a complex, unshared infrastructure to connect the packaged infrastructure to the business logic. Understanding all those layers is a tremendous task.

Leading practice organizations are dividing this task in two, as well as creating a class of infrastructure developers responsible for all infrastructure issues. This allows application developers to concentrate on understanding the business needs and developing the appropriate code. This organization is more common today in Europe than in the United States.

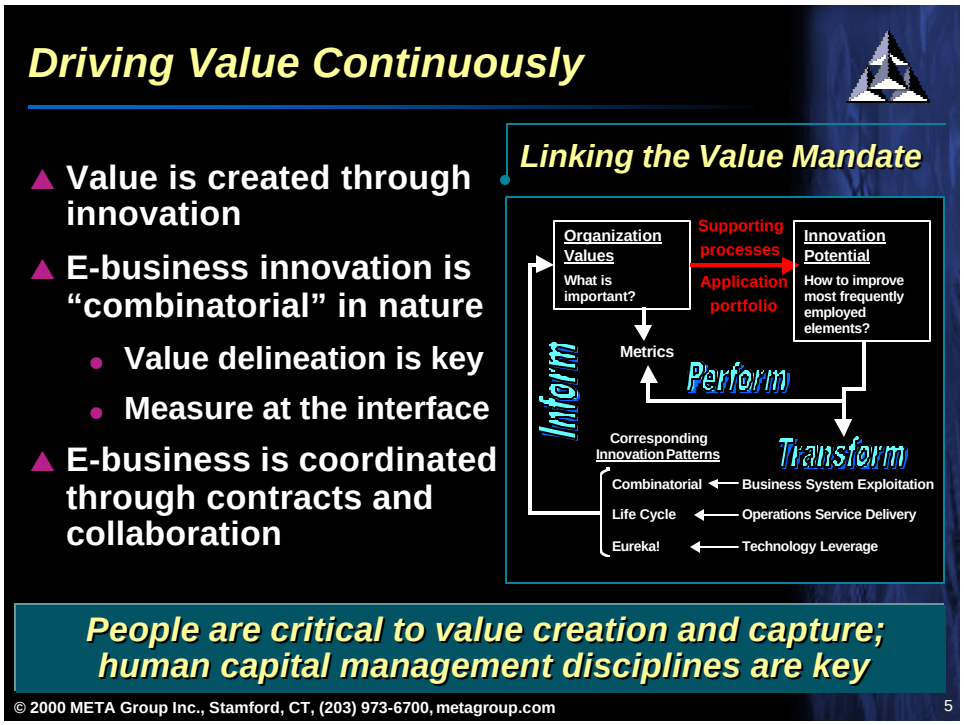
Training small groups of specialists is faster than trying to create a large body of generalists. It allows staff members to concentrate on doing a specific job better, and it makes retaining faster and easier as sets of skills become unneeded, helping to create a more agile staff.

Bottom Line: Capturing Business Leadership

Ultimately, the strategies outlined in this paper for achieving technical agility become the building blocks for achieving business leadership in the electronic economy. Leading businesses – whether they are dot-com start-ups or long established companies – are achieving leadership by being the first to leverage technical advances to change the rules in their markets in their favor.

In our Value Management Model (see Figure 2), META Group has defined three executive activities that firms must use to create, manage, and absorb business innovation: transform, perform, and inform. Of these, the first governs the actual creation and introduction of new business strategies, often based on technology, that can give businesses a commanding advantage.

Figure 2 – Value Management Model



Creating transformation does not require that companies develop new technology themselves, however. The transformational activities that provide the highest rate of business return are the strategies that combine mature business or technology components into new operational models. For example, IP, GUIs and file formats were all mature standards when they were combined to create the Web.

Agility, therefore, requires a new strategy, and more than that new ways of thinking – solutions built from independent components in place of tightly integrated applications, or multiple, linked databases in place of a single, central data warehouse. Sometimes these approaches are directly counter to traditional rules of system development, rules that were designed to conserve computer resources. Architecting for agility to provide flexibility, leveraging agile technologies for scalability, and organizing around critical skills will be key components of success in an increasingly chaotic new economy, and will be the hallmarks of tomorrow's market leaders.