Abstract

This paper briefly reviews the state of Enterprise System Architecting (ESA) and concludes that assumptions of incremental technical evolution (overlaid by selective organizational changes such as mergers or business growth/failure) are a driving force for ESA design. The paper proposes that ESAs evolve through a punctuated view of development – that new capabilities arise because of environmental, not primarily technical, drivers. It proposes a new model for ESA capability delivery – the service migration model - that no longer locates and delivers data to a processing site, but rather, as data sets reach the hundreds of gigabytes to the terabyte range, delivers services to the data set sites. It discusses the issues and implications for management decisions regarding ESA choices.

1. Introduction

We define an Enterprise System Architecture (ESA) as a structural and operational framework consisting of hardware, software, communications, data and information, and people, processes, and places, for developing, implementing, and executing business applications using a “system of systems” approach. We use the term ‘business’ here in the broadest sense to indicate the functional operations of any organization – private industry, educational institution/non-profit entity, or government agency – to achieve their mission.

The challenges faced by ESAs and requirements for ESA responses to the challenges have been addressed from a variety of perspectives. [6] We have reviewed the state of ESAs seeking to address these challenges from a process and pragmatic perspective using a historical view based on the assumption of incremental ESA technical evolution (overlaid by selective organizational changes such as mergers or business growth/failure) as a driving force for ESA design. We believe, however, that ESAs may well evolve through a punctuated view of development – that new capabilities arise because of environmental, not necessarily technical, drivers. As a result, we see an ESA as a highly complex socio-technical ecosystem rather than a monolithic environment.

This view leads us to suggest a number of research questions to be pursued to test and validate this view. Further, observations of the changing socio-technical environment of today lead us to propose a new model for ESA capability delivery – the service migration model. To date, architectures have been focused on locating or delivering data to a processing site. But, as data set volumes reach the hundreds of gigabytes to the petabytes range, it will be more functionally effective, less disruptive to the organization and systems, and more cost efficient to deliver services to the data set sites. The proposed approach raises a new set of challenges and issues for the enterprise system architect as services become commodities and fees are imposed for transmission of large quantities of data. The realization and wide adoption of this model will require new technical infrastructure and policy at both the client and server systems.

1.1 A Brief overview of ESA

Armour, Kaisler, and Liu [1] defined an architecture as a set of components and the relationships between them. A system architecture encompasses, as described by [2,3], a variety of components: software, hardware, communications, databases, people, places, and processes. An enterprise is often ill-defined, but we take it to be a large, complex ‘organization of organizations’, e.g., a large company with many different divisions or units each responsible for achieving its own mission, but contributing to the overall enterprise mission. Think of, for example, IBM, Microsoft, General Electric, or the Department of the Treasury as enterprises. An ESA is a ‘system of systems’ where the ESA incorporates many system architectures distributed across the enterprise. Complexity arises because a system architecture may support multiple units of an enterprise although some units may have their own dedicated system architectures. Problems arise, for example, during mergers when disparate system
architectures must be integrated or even merged into a holistic enterprise system architecture.

1.2 A Punctuated View of ESA Development

The research and descriptive literatures on ESAs focuses heavily on descriptive, technical, and analytical assessments of the relative advantages of various ESA models. The literature has made few assessments of the macro level environmental forces driving ESA component development and adoption. When macro level forces are considered, they are addressed as individual forces (such as a merger or business unit purchase or sale) and generate a subsequent need to (locally) optimize the architecture over some managerially defined time period. Casual observations regarding the macro forces affecting ESAs seem to assert that ESA component developments are primarily driven by relatively strong (and often steady) technological improvements that enable overall rapid increase in ESA capabilities. Rational decision making assumes that organizational adoption of new structures and processes will occur under stable conditions or those embodying organizational shifts will be driven by benefits models that seek to capture long term benefits at organizational levels by minimizing costs and maximizing benefits.

We observe that ESAs are often described as developing in an evolutionary manner as a result of growth in technical capabilities and demands coupled with organizational changes. At various points in time – capabilities may jump ahead of demand, cost may decrease precipitously, or new demands may pop up. However, the dominant description of ESAs is one of an incremental nature – gradual evolutionary change and steady growth in demand for services, storage, usage, and new services.

1.2.1 One View of ESA Evolution

Over the past five decades, information technology has evolved at a rapid rate. A rough characterization of technology based on where information is processed is presented in a table 1. Historically, data has moved to the processing or service system, if it does not already reside there, and results are returned to the user. For example, the user sends a query via a Web browser that retrieves a web page. The description of the web pages is transmitted in HTML, but is rendered at the user’s terminal – whether workstation or mobile phone – by local code.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Processing</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mainframe</td>
<td>No Physical Distribution</td>
<td>Static Physical Centralization</td>
</tr>
<tr>
<td>PC/Workstation - LAN</td>
<td>Static Physical Distribution</td>
<td>Static Physical Centralization</td>
</tr>
<tr>
<td>Workstation - WAN</td>
<td>Static Physical Distribution</td>
<td>Static Physical Decentralization</td>
</tr>
<tr>
<td>Workstation/Server -Web</td>
<td>Static Physical Distribution</td>
<td>Dynamic Virtual Decentralization</td>
</tr>
<tr>
<td>Mobile/Wireless</td>
<td>Dynamic Physical Distribution</td>
<td>Dynamic Virtual Decentralization</td>
</tr>
</tbody>
</table>

These models worked and work well when the amount of data to be moved was small compared to the amount of processing (e.g., the size of the code and number of instructions executed) to be done. However, the demand environment is dramatically changed today. Enormous complex data sets, detailed graphics, and audio/video files are rapidly becoming a significant portion of if not the primary information transferred. Many have noted that a communication network’s performance begins to degrade as it is overwhelmed by massive amounts of data. For example, some Internet service providers are considering charging customers per gigabyte of data downloaded in addition to the basic connection and service fee (for example, Time Warner Cables’ test in Beaumont, TX).

1.2.2 ESA Design

How organizations strategize or plan for this is of critical importance. An ESA design for an organization, large enterprise, or government agency is often more dependent upon a rational analysis of the business requirements used to realize enterprise goals rather than of the components of the ESA itself. The unstated assumption is that an organization’s ESA will have relatively stable or improving technical capabilities (lowering or known costs), and that the time-phased organizational demands for these specific capabilities will be the drivers for an alternative analysis, assessment, and decision regarding the appropriate ESA for the organization.
If one assumes that ESA design decisions are made based on somewhat simplified assumptions about increasing capabilities (Moore’s Law\(^1\)) and incrementally evolving technology, observations about service demands, and assumptions about costs: a significant research question can be posed concerning ESA decisions and the attributes of ESA components. In simple terms – what architectural choices may change, what ESA may become quickly dominant next; and how can we learn what assumptions regarding ESA drivers may predispose decisions or hide coming trends?

### 1.2.3 Challenging the ESA Design Assumption

A recent report on Ultra Large-Scale Systems, prepared by the DOD Software Engineering Institute [9], suggests that we view large-scale systems as socio-technical ecosystems. An ecosystem comprises a dynamic population of interdependent entities embedded, perhaps transiently, in a complex and changing environment. This correlates well with the view of architectural frameworks presented by [1,2,3], Zachmann, The Open Group, and other entities, which espoused a critical human element to ESA design.

Observations about great cost changes, technology shifts impacting capacities or capabilities, or new functional requirements necessitating unimagined service changes, may make a punctuated view of ESA development and adoption a better description. Such changes necessitate an understanding of the specific environmental assumptions underlying an ESA.

An example from the field of paleontology is offered to support this possibility for ESA change. For many years in the late 19th and through a large portion of the 20th century, the process of natural evolution was assumed to be explained by a gradual process of evolutionary development (and selective in-breeding), growth (or demise of species through competition), and climate changes. However, in the last quarter of the 20th century, it has been strongly suggested that another model of evolution is not only possible but highly probable. This argument is that catastrophic change in environment may well cause changes in the evolution of one or many species. Simply stated, some species do not grow, evolve or decline gradually over eons, but are driven to extinction by random events. Further, this new (and clear environmental landscape) gives other species the opportunity to rapidly grow and achieve a dominance in the environment.

Over the past year (2007 to 2008), the price of gas at the pump has increased by almost 33%. This has forced major disruptive changes in the way people and organizations conduct business. To cite a few examples: people are driving less and switching away from so-called “gas-guzzlers” or using public transportation more, when it is available. Airlines, experiencing significantly increased fuel prices, attempt to make up the difference by charging for hitherto free services, such as baggage checking fees, raising ticket prices, and eliminating free snacks and meals. Analogously, we expect that data transmission fees will force organizations to adopt alternatives to bringing data and applications together.

### 1.2.4 Benefit of Understanding the Drivers of ESA Designs and Changes

The resulting improved understanding of ESA design has important and strategic implications for organizations. The results can strongly benefit the IT executive by providing an explanation of the assumptions that underlie the selection of an architecture, and the implications when risks or probabilities are high that changes in those assumptions may occur. Executives may therefore better understand what may happen – and, what contingencies should be considered. Conversely, the developer of the components can ascertain where they may direct their efforts to make a significant attack on the market or to capitalize on the potential for an ESA shift or change that is likely to be adopted. Once identified these “shearlines” between ESAs and their components provide clear boundaries within which risks and costs can better be defined and contained.

This work also presents an opportunity to advance our understanding of the drivers, development and environmental changes that affect the dominance of ESAs and test our understanding through prediction of coming changes. We suggest a research program that utilizes a punctuated ESA assumption model for examining the dominance ESAs. It suggests hypotheses for the reasons for ESA dominance, and can yield predictions of the coming acceptance or adoption of an ESA that is not currently in vogue.

---

1. [http://www.firstmonday.org/issues/issue7_11/tuomi/](http://www.firstmonday.org/issues/issue7_11/tuomi/) "It is important to note that even in its simplest form Moore's Law actually may be interpreted in several different ways. Moore's 1965 prediction that in 1975 the most cost efficient chips would incorporate 65,000 components was, in fact, wrong. The empirical support for Moore's Law strongly depends on two important factors. First, there exist many types of silicon chips and they have historically evolved differently. For example, the number of transistors has typically grown much faster on memory chips than on microprocessor chips. Second, each chip usually has a life cycle of about a decade, during which its manufacturing costs and selling price radically change.”
To illustrate our punctuated view of ESA development, we examine one system architectural framework (service-oriented architecture) and suggest a new model for service provision. In section 4, we describe some of the challenges associated with this model and the methods by which the challenges may well be met.

2. Service-Oriented Architecture

Service-oriented architectures (SOA), one type of ESA architectural design pattern, are a style of system architecture that packages business process operations into services that are comprised of a collection of software modules. These services implement common and specific business applications processing. By logically decomposing business process operations into discrete services, organizations can determine which operations are common to many operations and which are specific to a business process area. Common operations, built as independent services, can then be reused in many different business process areas. Examples of services include filling out an online application for an account, viewing an online bank statement, or placing an online booking or airline ticket order.

Software reusability has been one of the holy grails of software engineering and development for over two decades. However, previous attempts were focused on small components – typically composed of few function points (or tens to hundreds of lines of code). By comparison, an SOA’s atomic objects are usually 100 to 1000 times as large. Unlike components and modules, which invoked each other directly through synchronous procedure calls in a hierarchical or concurrent arrangement, services access each other through a set of asynchronous message-based exchange protocols. If such protocols are standardized, then one service may invoke another that can provide needed functionality without regard to whether it used that same service previously. SOAP has become the preferred communications protocol. A standardized mechanism for providing and describing data must be used to ensure the service can satisfy the diversity of potential clients. As of this writing, one of the most popular means for one or more services providing self-describing data structures for communication is XML.

The goal of SOA is to allow ad hoc collections of services to be (dynamically) combined to provide an application to some client. When the application need is met, the application bindings are dissolved. The presumed benefit of the SOA approach is that the marginal cost of creating the nth application is near zero since, presumably, all the services and binding agents already exist.

Figure 1-1. Simple SOA

2.1 Rethinking the SOA Model

As an example of applying our view of punctuated evolution in ESA design and implementation, we consider a new approach to the SOA model for system architectures. First, let us review the current approaches to SOA, which can be generally grouped into three categories:

- **Internally hosted cloud**: Services are built and a directory of available services is maintained within a single domain such as a corporate directory. These services are the products of other separate development efforts resulting in specialized services that expose complex organizationally-specific functionality. The services are intended to be consumed by a specific thin or thick client but can be customized or extended by request.

- **Third party hosted cloud**: Services are built independently of the end user and largely independent of each other. Services tend toward generalization and are likely to be either too heavy or too light for any given consumer. An individual service will make up part, but not a majority, of the consumer’s solution. The hosting organization accepts responsibility for service availability and scaling the services to meet demand.

- **Software plus Services**: Services are intended to augment rich client applications generally adding functionality that connects users or federates queries or distributes processing. Applications operate without services reducing risk but not removing the hoster’s need to support SLAs and scale services to meet demand.

While oriented toward different business models, each of the three approaches noted above requires
data to be transmitted to the service location for processing and the return of processed results to the consumer. Obviously, this capability requires significant environmental and technical support, equipment, planning and scheduling resources, and transmission capacity. All must be provided within a “reasonable “and known cost model.

2.2 A New SOA Approach

We propose a new approach whereby services migrate from one processor where service code resides to processors where data needs to be processed. The service migration concept operates as follows. A large amount of data, such as a large data set of gigabytes to terabytes, needs to be graphically rendered. Rather than ship the data to a system to have it analyzed and rendered with the resulting images returned to the data source, the data owner desires to acquire a service comprising, perhaps, a few megabytes, that comes to his system to analyze the data, render it, and display it. The data owner usually does not wish to purchase the service, only to acquire the use of the service for the time necessary to analyze and render his data. Once the data is processed, the service is removed from the data owner’s system. The tradeoff is the transport of a few megabytes of code versus many gigabytes of data. The obvious benefit is that communication channels are not overwhelmed with data by a single consumer. If multiple data sets need to be rendered, then the reduction in communications is substantially greater.

Code delivery is not a new concept, but has been utilized to date for very specific applications - scripts (JavaScript, VBScript), Java applets, ActiveX controls, Flash animations, and macros embedded within Office documents – to name a few. In these implementations, the user is not aware that mobile code is downloading and executing in their workstation. Moreover, the amount of code that is downloaded is on the order of a few thousand bytes. In our approach the user initiates the identification, acquisition, and delivery of the service.

This model is conceptually simple and has many historical analogues. For example, doctors used to make house calls to patients. At one time, a man used to come around the neighborhood and sharpen knives and other cutting instruments with a tool strapped to his back. A housewife calls Merry Maids to come clean the house. Today, temporary workers are “rented” to provide different types of services in offices and at job sites. An organization can hire clerical and administrative personnel, programmers, janitors, and even Chief Executive Officers on a temporary basis. Each of these individuals carried the tools of his trade with him – knowledge, instruments, and skills. Each of these individuals arrived at a location without foreknowledge of the situation he would find, the scope of the task, and the level of effort required. Most charged a fixed rate per hour or per unit of work, e.g., sharpening one kitchen knife for $0.50.

The idea of service migration is an intuitively appealing idea, particularly in an environment where a user might be charged for communications bandwidth based on the number of bytes transferred, and incur significant additional charges for other storage services, and necessary support. Whereas process migration attempts to balance load across a set of platforms that are computationally heavily loaded, service migration in the context of very large data sets attempts to balance data transfer volumes across heavily loaded communication networks.

2.2.1 Precursor: Process Migration

Process migration [8] is the relocation of a process from its current location to another location. Processes can be migrated before they start execution (non-preemptive migration) or, in certain systems, during execution (preemptive migration). The latter operation requires moving the process, the data, and its environment and description to the new site and removing it from the old site.

Processes have been migrated in tightly-coupled symmetric multiprocessor systems to achieve load balancing in order to reduce the execution time of parallel applications and improve the usage of computational resources. However, this has normally occurred in a homogenous operating system environment. Processes are usually part of a large application that has distributed its work across the multiple processors of the system. Process migration is under the control of the operating system, which determines when and where processes migrate in order to balance load and maintain overall system efficiency and application performance. In general, process migration is constrained by (1) a requirement for environmental homogeneity and (2) subservience to operating system decisions.

2.2.2 Precursor: Software Viruses

Historically, the observed (coming) ESA change has been happening for many years, but on a very small scale. Software viruses, which can self-replicate, migrate from system to system in order to infect as large a set of systems as possible. Regardless of their motivation, virus migration is usually opportunistic as opposed to seeking systems with specific characteristics. (This would be an important characteristic of the processes in the new
ESA model.) In general, software virus migration is constrained by (1) homogeneity and (2) opportunistic access to the target system, both readily available attributes for large data sets requesting specific processor service.

### 2.3 Emerging Model

The proposed ESA model will possess a number of other important characteristics. Analysis of the services required will be used to describe it more completely. For this discussion, a service is an integrated application that provides specific functionality to a client. The service provides an API that, among other things:

- Provides a mechanism by which it receives data from the client and emits data to the client
- Provides a mechanism for broadcasting specified attributes and characteristics of the service.
- May provide a mechanism for self-configuration at the client site.

Service migration involves the transport of the service application from a provider to a client for a specified duration. However, numerous issues and challenges must be resolved in order to implement and utilize services within this service migration model. The client’s environment must provide certain capabilities and resources to support the service. And, a service migration protocol needs to be established to facilitate service acquisition, installation, operation, and eviction. The following sections describe some of the critical issues that must be addressed by developers and users.

### 3. Acquisition Issues

Today, a user has a choice of several methods for acquiring software: purchase a Commercial Off The Shelf (COTS) software package, have a software package custom-designed, or send data to another site for processing and have the results returned. In the first two cases, the software is installed on one or more local platforms. In the latter case, the data leaves the owner’s direct control putting it at risk.

In a service migration model, additional acquisition methods can be made available, such as open bidding, ordering from a menu, or scheduled delivery. Our model differs in that the software is sent to the user for installation for a fixed duration, a fixed number of processing operations, or even a fixed type of data. This approach can be viewed as similar to downloading software on a trial basis with limited functionality, but where continuing usage is contingent upon purchasing a key from the software provider. Acquiring the key unlocks the software’s full functionality and converts it from a trial basis to a full purchase. Thereafter, the user is permitted to use the software as often and in as many ways as he/she sees fit. However, in the service migration model, a mechanism must be provided to evict the service from the client’s platform upon the expiration of the contract.

The intellectual property rights of the software owner need to be protected as he/she is leasing the software to the user under fixed conditions. To implement this model will require additional infrastructure to enforce the terms of the service contract.

### 4. Implementation Issues

Implementing a services-on-demand ESA model in an enterprise will require the enterprise to implement clear policy, governance, and legal precedents as well as a runtime software infrastructure supporting the various vendors/suppliers of services. Establishing these environmental conditions is very costly and time consuming. Further, political processes and policy changes may pose significant barriers to long term decision making in these areas. The overhead alone of doing this makes it prohibitive for an organization to acquire services on a “piece by piece” basis for the many components in an ESA. What might be advantageous would be a component enabled with the data to “know” the conditions are all met so it can participate in an ESA.

#### 4.1 Infrastructure Support

In order to enforce the service contract, and enable the component to know it meets participative conditions, additional infrastructure will be required. This infrastructure requires a subsystem that:

- Exposes candidate services
- Enables the transaction between provider and consumer to be pre-defined, limited if necessary, and documented
- Applies the service to in accordance with the consumers operational context
- Manages the services life-time including clean-up and removal.

Additionally, it may provide performance feedback and utilization information to the service owner.

#### 4.2 Computing Environment Integration and Eviction
The service application will be temporarily integrated into the user’s environment in order to provide its functionality. Aside from the security issues, several issues must be addressed.

4.2.1 Environment Knowledge

A service application must incorporate assumptions about the nature of the environment into which it will be integrated. These assumptions need to be accessible through querying prior to contracting for the service or provided as part of the initial service contract, perhaps through the service acquisition protocol. The client, as part of its selection process, needs to know what the service’s requirements are to operate correctly and efficiently at the client’s site.

If the service is self-configurable, then it needs access to an API that allows it to obtain knowledge about its environment and the constraints the environment will impose upon it. Once configured for the host, the service must also exclude operation within an environment other than the one originally contracted. This would prevent the client, for example, copying the service to another processor to use it.

4.2.2 Installation and Integration

Installation and integration encompasses the activities related to accessing a communications endpoint, gaining and leveraging the necessary level of access to operate, persisting the service to the appropriate user store, executing one-time or pre-execution actions either directly on the host operating environment or via a previously installed service support service, pre-execution validation, and exposing the running service to the host monitoring subsystems.

How does a service dynamically link into a user’s computing environment? Downloading a service onto a platform is relatively easy. Given the infrastructure mentioned above, it can be invoked using standard OS functions, monitored, and erased when the service contract is fulfilled. Provision of file names and database schemas needs to be resolved.

4.2.3 Service Eviction

Service eviction has two connotations – normal and abnormal termination – but both lead to the same issues. When a service is to be removed from a computer system, the operating system at the site must be able to locate and remove all components of the service. We will assume here a service that is not malicious. From a security perspective, no remnant of the service can remain once it has been evicted from the system.

When a service terminates normally, it should behave in a consistent and predictable manner. It should respect the principles and policies that are implemented on the platform within which it is operating. Additionally, a terminating service should always invoke its runtime framework, service coordinator, and, as necessary, engage the service user to ensure the sequence of shutdown activities acquiesces to other running services.

When a service terminates abnormally – either due to an error condition or because of OS intervention, the service should follow the well known and agreed upon sequence of operations as defined by the service contract. This may include but not necessarily limited to; sudden termination with data loss, timed termination with user intervention, user informed and termination delayed until the user intervenes, and state-safe partial writes before termination completes.

4.3 Dynamically Changing Needs

ESA approaches and targeted COTS-based implementations tend to drive system and application solutions along divergent paths. ESA focuses on satisfying a well defined user community which requires dynamically changing business needs and services that are constantly aligned with changing business requirements. COTS solutions, intended to be used by a very large unknown user community, often offer more functional capability than is required, but which an organization must accept to obtain the functionality it requires. There is a concomitant cost for this additional functionality, and frequently a cost of modifying the COTS to some extent to meet user needs.

Dynamic service composition and delivery would allow a service to be assembled and delivered to meet specific fine grain requirements of the intended user as well as expand or contract the available services on-demand.

5. Security Issues

Security issues that arise from the dynamic nature of migrating service can be addressed by making the ESA and its services adhere to the basic Bell-La Padula axioms of computer security.

This means the user must be assured that the downloaded service uses a combination of pre-execution, runtime, and post execution directive. Prior to execution a service contract might require the user to be provided or added to a membership list by an administrator for either the local infrastructure or the service provider or both. As the service loads, its
authenticity can be verified via certificate or key exchange. At runtime the service must comply with local execution policies and emit a local system visible trace for logging. Post execution the service should provide user visible signatures / certificates / keys for return data to be, independent of the service, verified for source and manipulation.

If the service provider “pushes” the service to the client, the service provider system has to be identified to the Client’s security system in order to allow the service application code to pass through the firewall and other security systems, including antivirus and spyware checking programs.

Although the client and the service provider have signed a contract, the service provider and client must still be authenticated to each other in order to complete the transaction. Depending on whether the service application code is “pushed” or “pulled”, the service provider or client must be approved to access the other’s system.

6. Self Reporting of the service to the Owner

Owner equities and protections will be important parts of the ESA described in this example. For control (and possible billing verification or validation), a service is likely to report back to its owner the duration or number of times its been used (or some other similar metric). Client and owner must agree in the contract on what is reported, how often, and what mechanisms are put in place to prevent leakage of information from the client system.

7. Valuation Issues

The ESA proposed will have a complex valuation (and related billing) model that varies depending upon the characteristics of the consumer. What this means is that service value is based on return-on-investment schemes and will vary wildly from one consumer to the next. Small, single-use consumers of a service may not recover the cost associated with accessing a service, while a larger consumer might find leveraging the same service repeatedly reduces direct operating costs. As services are used, feedback from consumers could form a baseline of value derived. This would require a grace period where initial or first time consumers of a service have no reference point for the service’s value.

As ESA software services evolve, it is possible that service creation will lead to a commodity market in certain types of services, such as order entry, online reservations, and certain kinds of financial analysis. These readily understood, and more openly available services will be supplemented over time with more unique and complex services such as data mining, intelligent querying of data bases, and intelligent agents.

7.1 Service Pricing

A critical issue will be how a service is priced and how one can measure the functionality offered versus the price demanded. A diverse range of options exist here.

Typically, COTS software is assigned a price based on attention to what the market is likely to pay and the pricing of competitive products. Incremental enhancements, now known as service packs, may be free or low cost. Subsequent upgrades and enhancements require new purchases (e.g., moving from Windows XP to Windows Vista) with repricing based on a re-evaluation of the market.

This model can be viewed as similar to the NetFlix model which mails DVDs to the client for a fixed rental price for a fixed period of time. During that period, the client can view the DVD as many times as he wishes.

Custom-designed software, such as the government often procures, is usually acquired through a cost-plus-fixed-fee contract (CPFF) with the support costs priced separately and often exceeding the initial development cost. If the service provider configures a unique service based on user specifications, pricing may be fixed or varied depending on the amount of effort necessary to configure the service prior to delivery.

7.2 Updates and Enhancements

Suppose an organization acquires access to a service for a fixed period. Within that period, one or more updates or service packs are made available for that service. One issues arises is whether the user is entitled to the update and at what cost. A simple answer might be that if the service is acquired for a short duration, the next acquisition of the service would just retrieve an updated version of the service. Longer term contracts might include the update for free or include an optional fee allowing the user to obtain the update, if he wants.

8. Legal Issues

Numerous legal issues affect the service delivery model. We have separated these into issues addressing contracting, standards, and information transport.
Service delivery should require a contract between the service owner and the service user. Service acquisition can be affected via different models depending on the nature of both the owner and the user. For example, if the service owner is a commercial firm and the user is the Federal government, in most cases the government is required to conduct a fair and open solicitation to acquire the service.

Service contracts may be directly affected by the initial provider/consumer relationship. Some commodity services may be created by one organization and consumed by another. These will follow the more traditional direct and brokered contract formats. Other services will be developed within an enterprise and consumed by another business unit or division of the same enterprise. Contracts will most likely reflect cost recovery or direct expense models.

There are numerous mechanisms for electronic payment for services rendered. Two models of particular interest to the service migration model are PayPal and eBay. In both, the client establishes an account for automatic payment upon acknowledgement of a contract. Once the account is debited, the service is transferred to the client. Both of these methods incorporate mechanisms for disputes including nondelivery and quality of service.

8.1 Information Transport Costs and Controls

Services implemented in software represent a capability that is likely to be subject to taxation. If not in the United States, certainly overseas, such as the European Union’s Value-Added Tax. The application code for services may be treated as a manufactured object, although clearly it is not quite. Transport of service application code across national boundaries raises some interesting legal problems that will have to be thrashed out in the coming years. For example, can service application code transit national borders in a manner similar to an airplane flying through a country’s national airspace?

In the United States, many businesses already make use of software services located outside our boundaries. Today, it is largely restricted to shipping data overseas. But, technical progress has encouraged fragmentation within the service sector. In the future, as certain countries develop indigenous software development capabilities at cheaper cost than companies in the US, companies may find it (much) cheaper to import the services rather than export the data. Since data/information is a fundamental asset of many corporations, this will serve to protect corporate information, but at the expense of purchasing services from foreign providers.

Many countries restrict the export of data, even if it belongs to multinational corporations, beyond their boundaries. And, many countries are seeing the transport of services as subject to tariff(s) just like tangible goods.

9. Assessing Utility, ROI, and Risk

The table below summarizes the attributes of a service migration ESA and the issues and needs they address.

<table>
<thead>
<tr>
<th>Service Migration ESA Attribute</th>
<th>Addresses Issue or Environmental Need-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API w/ data receipt/emit; attribute broadcasting; list dependencies</td>
<td>Consumer opt-in and multi-platform support</td>
</tr>
<tr>
<td>Exposes Candidate Services</td>
<td>Service Discovery and automatic negotiation</td>
</tr>
<tr>
<td>Predefined, Limited Transaction</td>
<td>Commodity based scaling serving multiple user types</td>
</tr>
<tr>
<td>Adapts to Operational Context</td>
<td>Complies with local policies and other services</td>
</tr>
<tr>
<td>Service Eviction Capability manages Clean-up and Removal</td>
<td>Graceful exit and termination under various conditions</td>
</tr>
<tr>
<td>Possesses Knowledge of Integration Environment</td>
<td>Predictable and compliant execution</td>
</tr>
<tr>
<td>Self Configurable</td>
<td>Leverages local resources in a platform compatible manner</td>
</tr>
<tr>
<td>Dynamic Service Composition and Delivery</td>
<td>Load and configure in response to user demand</td>
</tr>
<tr>
<td>Security Execution Directives</td>
<td>Consistent reliable auditable logging and local security policy compliance</td>
</tr>
</tbody>
</table>

The business utility of on-demand services is hypothesized to be high because on-demand acquisition of data analysis and processing services is likely to increase organizational flexibility. This is an
important characteristic in an age of dynamic change, mergers, outsourcing, and extensive competition. Utility will also increase because exposed services provide otherwise walled off information to users with rights to the service and its underlying data. Organizationally-controlled libraries of composable application services would strengthen reuse and cross-organizational sharing of processes, limit N+2 reuse costs. Finally, user experimentation will be more frequently possible with vendors offering no-cost trials or limited usage, low fee experiments that will encourage innovation in IT and in the design of programs for services.

Controls for ROI, and cost-benefit analysis will be strengthened because large single vendor sources will not be mandated. Instead, service consumers can assemble large complex solutions organically by procuring many small and relatively inexpensive services a few at a time. This spreads the spending plan for the enterprise over time and across multiple suppliers and directly supports incremental development approaches.

Enterprises using assimilated ESAs composed of multiple services will reduces risks as they transfer the complexities associated with maintaining a highly available hosting environment to external partners who would specialize in that line of business, thus limiting their own risks.

10. Future Work and Conclusions

We have suggested a new model for providing services within enterprise system architecture. Exploration of this new model requires a theoretical analysis of the economic and business utility issues; a functional analysis to determine the types of services amenable to this model; and a prototype system to explore the design, development, and implementation of the infrastructure, the services, and their execution.

We have briefly described a new approach to an ESA: a service migration concept that extends the notion of a service-oriented architecture. We have address several, but not all, of the challenges that can affect the implementation of such an approach and suggested some solutions. Our intent is to open a dialogue on this approach and solicit feedback from interested parties. We are developing a research program to explore some of the ideas expressed here and welcome collaboration from other interested parties.

11. References


