The Tao of e-Business Services

Steve Burbeck
Emerging Technologies, IBM Software Group

Executive Summary

e-Business services, which are loosely coupled computing tasks communicating via the Internet, play a growing part in business-to-business (B2B) interactions. Traditional computing tasks, such as database access or commercial transaction systems, are being wrappered as services and connected to the Internet at a rapid pace. And new tasks, such as computerized auctions and e-marketplaces, are being created as business services. Simply put, e-business will be based on a service-oriented model.

We argue that, to work easily, flexibly and well together, services must be based on shared organizing principles that constitute a service-oriented architecture (SOA). We reserve the term "service oriented" for architectures that focus on how services are described and organized to support their dynamic, automated discovery and use. We do not address systems based on manually hardwired interactions such as those used in EDI systems.

For dynamic automated discovery of appropriate services to be practical, the collection of available services must be organized into computer accessible category hierarchies (i.e., taxonomies) based upon what the services in each category do and how they can be invoked. We argue that these taxonomies will be maintained and made available by categorization services, or brokers, analogous to Yahoo or Netscape Open Directory. If the growth of Yahoo is any example, the provision of category servers will be a business opportunity in itself. Thus we expect B2B entrepreneurs to provide categorization services for various types of B2B services, and for various industries.

Each component in a service-oriented architecture can play one (or more) of three roles:

- service providers publish the availability of their services,
- service brokers register and categorize published services and provide search services.
- service requesters use broker services to find a needed service and then employ that service.

Service Descriptions, in a standard XML format, are associated with each service. These service descriptions are key to all three roles in that they provide the information needed to categorize, choose, and invoke an e-business service.

This white paper explores the implications of the widespread availability and interaction of e-business services. We discuss the design-time organization of e-business services, the run-time organizational issues, and the economics of the business models that are implied by the interactions between service providers, service requesters and service brokers.

Introduction

A world in which myriad e-business services connect and collaborate with one another over the Internet is fast becoming a reality. Business-to-business (B2B) service interactions already exist using a variety of schemes that range from very rigid point-to-point EDI interactions to open Web auctions, e.g., Ariba, Chemdex, eSteel, and eBay. Thousands of businesses have already made some of their IT services available to their customers or partners on the Web. Most of these services are intended for use from a browser. But with technology such as WIDL from webMethods (www.webmethods.com), many of the Web-enabled services can also participate in B2B collaborations.

Some e-business services, such as stock tickers or product catalogs, simply provide information. Other e-business services enable light-weight commerce such as B2B purchasing of office supplies or support mission-critical B2B commerce transactions such as multi-million dollar purchases of CPU chips by a PC manufacturer. Various e-business services also represent different fundamental business models [see, for example, "B2B E-Commerce Hubs: Towards a Taxonomy of Business Models," Steven Kaplan and Mohanbir Sawhney, Harvard Business Review, 2000].

Today, services are collaborating without any overarching vision or architecture. Techniques for B2B collaboration vary from one case to another. If we are patient, consistent well-understood conventions will eventually evolve out of trial-and-error techniques. However, trial-and-error is messy, painful and slow. We would prefer a generally accepted unifying architecture that makes it clear to IT architects and business system designers how the various services should work together.

Service architectures have been proposed by various companies. The best known are Microsoft's Biztalk (and/or SOAP) initiatives, HP's e-speak, and Sun's Jini. We believe that these efforts lack sufficient generality to cause a worldwide community of services to coalesce. Each of these vendors envisions the entire world of interacting services depending upon technology from the respective vendor. Sun presupposes that everyone will use Java, Microsoft wants everyone to use Windows, and HP imagines everyone using their e-speak core. Meanwhile, the world remains stubbornly multi-vendor, multi-language, and multi-OS. IBM embraces that diversity.

Efforts to establish a vendor neutral architectural approach are the focus of the CommerceNet eCo Architecture for Electronic Commerce Interoperability. The eCo working group includes participants from a wide range of businesses interested in eCommerce, including IBM, Sun, and Microsoft. As a result, its intent is to explore ways to describe e-business in XML terms rather than to compete with initiatives like Jini, Biztalk, or e-speak. While the eCo architecture shares some of the technical features of what we discuss here, it lacks many of the features that we propose to support dynamic lookup and discovery of services at runtime.

This document is based on a view of service collaboration that is independent of specific programming languages or operating systems. Instead it relies on already existing transport technologies (such as the Internet and HTTP) and industry-standard data encoding techniques (XML). The unifying element proposed here is a way for services to be described and categorized. These descriptions can be created and published by service providers, can be categorized and searched by broker services specialized for that purpose, and can be found and used to invoke the services by requesters of the services. Thus there are three roles: service provider, service requester, and service broker with the relationships shown in the following figure.

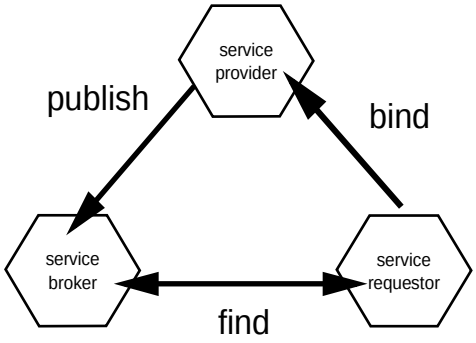

Figure 1. Service roles and interactions

Collaborating services resemble a theater production. After appropriate script choice, casting, set design and rehearsals, actors are brought on-stage to play their roles. Similarly, collaborating B2B services are carefully scripted to achieve the desired result. The cast of specific services must be chosen, then rehearsed (i.e., tested and modified until they are "right"), and finally released for public performance. Each of these issues must be considered somewhat independently. Business needs largely determine the basic script of the eventual "performance". IT architects and designers refine the script and either specify the cast (in a static design) or the casting rules (when choice of services is made dynamically at runtime). Finally, the services actually put on the play at the behest of a customer. The purpose of a Service-Oriented Architecture (SOA) is to specify the conventions and practices to be followed at the time the scripts for service collaborations are designed and at the time they execute. A Service-Oriented Architecture elucidates the mechanisms and practices by which the collaborators are specified. That is, what types of services will play what roles at runtime, and how services suitable to play each role are to be found. The architecture should also set forth the mechanisms whereby the selected services actually collaborate at run-time (i.e., communicate and negotiate with one another to obtain the appropriate results).

The flexibility of e-business services

The Internet is the fundamental new factor that has fueled innovation in business models and created an "arms race". Arms races are characterized by a positive feedback loop in which the "arms" create the very problem for which more arms are the solution. While first-to-market is not always a guarantee of success, the speed with which a business can adopt new business models and their technical underpinnings is certainly an important factor in success. Details of a service-oriented architecture will determine how flexible B2B services can be and how rapidly they can change, hence how powerful a factor they will be in the e-business arms race. Thus we wish an architecture that promotes flexibility and rapid change.

Consider three different usage scenarios for how B2B services collaborate at runtime:

1. Hardwired binding at design time -- The application "knows" the precise collaborating service because it has been hardwired during design. Therefore, the application also knows exactly how it will interact with the service.
2. Dynamic binding to a static choice of collaborator -- The application knows how to ask a broker for the precise collaborating service, because the designer encoded a specific enough query to pass to the broker. However details of the interaction, for example the choice of transport protocol or authentication, depend upon the service description returned by the broker at runtime.

3. Dynamic choice of collaborator -- The application knows the semantics and the APIs of the service to be used, but queries a broker with a search pattern that allows a set of alternative service providers to be returned. The application chooses from this list at runtime.

Today, largely because we are in the early stages of the growth of B2B services, most advocates of service-oriented architectures are focused upon the first two alternatives. After all, those who currently envision using or providing B2B services are taking enough of a leap into the unknown without dealing with a fully dynamic model. However, the third alternative provides the most flexibility and adaptability. We believe that the architecture must support all three alternatives.

Lessons from object-oriented architectures

Collaborating services on the Web resemble collaborating objects in an OO system, especially in a distributed OO system. So the lessons learned from two decades of experience with OO systems can help us to understand systems of collaborating services.

In the late eighties, when OO was rapidly gaining popularity, a number of languages that could not claim to be object oriented attempted to lay claim to the "object" mantle by asserting that they were "object based". They would claim that their languages could be used in ways that mimicked message sending or that they provided some form of encapsulation. Yet experience proved that most of the benefits of full OO systems came from the organizing power of class hierarchies rather than from encapsulation and message sending. Well-factored class libraries allowed OO developers to more easily design, build and maintain their systems. Without classes and inheritance to guide us, understanding the workings of the objects is much more complicated. Since these OO organizing principles were the very ones that object based systems left out, the claim to be object based quickly sank into well deserved obscurity.

The lesson to be learned from the failure of object-based systems is that the way services are described, organized, specified by potential users, and discovered amidst the clutter of the Internet will determine the success of B2B services. That is why we reserve the term *service oriented* for architectures that focus on how services are described and organized in a way that supports the dynamic discovery of appropriate services at runtime.

Architectural schemes that focus instead on service-to-service message protocols, i.e., on the details of how the various servers communicate rather than what they say to each other, should be characterized as *service based*. That characterization is not intended to imply that service-based architectures have no value. Within a single corporate system, where the entire system is under the control of one group, a service based approach can be used to break overly rigid legacy systems into collaborating services that provide dramatic improvements in flexibility and maintainability. However, service-based techniques alone do not scale beyond the span of control of an architecture group that defines and manages the semantic definitions of the services. This span is usually no larger than a single corporation.

The importance of semantics

The semantics of services, i.e., what they *do* and what data elements they manipulate *mean*, is the key issue. Business value results from B2B collaborations that do the right thing. If they do something else, the damage may be dramatic. How, then, do we trust that a service does the right thing before it is used? And how do we make that determination at Internet speeds?

In small-scale OO systems, interface compatibility usually implies semantic compatibility. That is, an object that implements the right set of messages with the right types of arguments probably does the "right thing". This is true in part because small-scale systems tend to be built by a small team of programmers with shared understanding of how the system operates, and in part because

small systems offer little opportunity for ambiguity. In large-scale OO systems the semantics provided by a given class cannot be reliably deduced from the message interface alone. While this may seem obvious, current proposals for service interaction, such as Jini and SOAP, assume that interface compatibility is sufficient.

The term *semantics*, as used here, refers to the meaning, in human and business terms, of the service, its arguments, side effects, and results. Even in the abstract, the semantics of a given category of e-business services is often much more complex than could be completely described by natural language text. Fortunately, business is comfortable with a certain level of ambiguity. Despite the long history of AI research, and even the efforts in the current AI spring thaw, we should be honest and acknowledge that only humans can determine or decipher software semantics. And ever-present software bugs prove that even humans do a poor job. Moreover, humans do it very slowly. At the sub-second time scales in which e-business services can collaborate, we require machine augmentation. One solution is to let humans decide which instances of services should talk to which other instances of services. That is, statically determine the collaborations at design time. Even then, there is the issue of ensuring that the service continues to do what it is supposed to do, i.e., hasn't been hacked, spoofed, diverted, etc. A second solution is to sort services into categories that determine the semantics much the way Yahoo! sorts Web pages into categories in their taxonomy. As with Web categorization, humans must create the categories and assert that a given service belongs in that category. Then at Internet speed, computers can test the members of the category, search categories, and the like. We primarily explore the second option here because the business value of B2B e-services depends on late binding, and flexible choice of collaborators.

Therefore, the aim of our proposal is to improve the semantic integrity of the categories and the services that occupy them. We rely on human judgement for the ultimate decision about what category of functionality is provided by a given service. But we propose mechanisms that allow automated B2B interactions to take advantage of the human categorization, with reasonable assurance of accuracy, at the speeds necessary for B2B interactions.

The need for security

Collaborating services are neither designed by nor policed by a single organization. Moreover, services may have substantial economic implications to the various businesses providing and using the services. Sensitive data must be protected. Perhaps even the existence of a service must be protected from unauthorized probing. And transactions must be enforceable. Thus, SOA must address issues of authentication, access control, encryption, non-repudiation, and authorization.

Organization principles for e-business service design

An economy of collaborating services will consist of many service providers and many service requesters. This economy will become powerful only when we provide organizing principles and structures, both at design-time and at run-time, to make the overall computing process understandable to individuals and to the social and business groups that need to embrace it.

Design is both a human and a machine issue. That is, organizing principles for analysis and design arise out of the interaction between technical organizing principles and the way people are most comfortable thinking about systems in terms of those technical organizing principles. For example, the invention of the subroutine (later generalized to the function) allowed programs to be subdivided for the first time into functional units. That technical innovation gave rise to the methodologies, analysis techniques, and design practices of functional decomposition. The

notion of a software object, which combined function and data in encapsulated units contributed new technical constructs such as classes, inheritance, and polymorphism. These innovations required new methodologies, analysis techniques, and design practices. At first, OO practices tended to look much like functional decomposition practices re-purposed and given new names. But it eventually became clear that OO design was fundamentally different. The most important task was to make good use of existing class libraries. Classes needed to be organized into well-factored class hierarchies and tools for browsing these class libraries were needed so that the libraries became easily accessible to OO designers and programmers.

Services represent yet another new form of computing that will require new organizing principles. Each individual service, unlike an individual function or object, is designed to satisfy a *business* agenda of an individual organization while collaborating with applications or services from other organizations. The organization of services must serve a constituency far larger than the technical professionals within a single company. Moreover, services, as a collection, depart dramatically from both functions and objects in that the set of services available on the Internet was never designed or booted and will never be redesigned or rebooted. The worldwide set of services available on the Internet grows and evolves organically. A simple description of available services is impractical both because of the scale of the Internet and because the set of available services changes from day to day or even minute to minute. The central organizational issue with services is how to organize and search for services when the dynamic economy of services is far beyond the purview of designers, either individually or as a group. The approach that works for function APIs is documentation of calls and argument types with natural language descriptions of semantics. The approach that works for OO systems is based on browsing class hierarchies. Both of these require too much human intervention and far more centralized control of APIs and class libraries than we can expect with B2B services.

Since the scale and dynamism of the set of available B2B services resembles the scale and dynamism of the Web, we should seek parallels with the problem of organizing and searching Web pages. There, categorization services, crawlers and search engines have emerged to handle the job. By analogy, we assert that the solution to the problem of providing organization for B2B services is that *the organization and categorization of services itself becomes a service available from multiple competing service providers*. Categorization services, may compete on the merits of their choice of taxonomy, on the up-to-date accuracy of their listings, and on auxiliary information such as quality-of-service data. These competing categorization services will arise out of economic or social “markets” in a manner similar to the way competing search engines and categorization sites such as Yahoo have emerged to organize Web pages. And, given the present market capitalization of Web search sites such as Yahoo or Excite, one can confidently predict that fortunes are waiting to be made by providing categorizations of B2B services.

These categorization services will be the brokers in Fig. 1 above. Service providers will publish, or list, their services with one or more brokers. The service provider knows the semantics of the service and therefore publishes it to the “right” category in the broker’s taxonomy. Service requesters, knowing the category of a service they need, ask the broker for a list of services in that category.

What is published, requested, and categorized about services is their *Service Description.* Service descriptions are XML documents that describe the semantics and the message API of the service. Describing the API of a service is relatively simple (use XML, named method calls, named arguments, and typed arguments). The description of the semantics begins with a human readable description of the service’s behavior. Both this human readable description and the API definition guide the placement of the service in the categorization service’s taxonomy hierarchy (see below). That placement, in turn, defines the semantics of the service for automated search and use by other services.

Categorization service taxonomies

To be useful as the fundamental organizing force in B2B services, the taxonomies created by each broker will need to meet requirements both for human usage and for use by machines.

- Taxonomies must be hierarchical to reflect commonalties larger than individual species. And, as is the case with biological taxonomies, there will be more than one way to categorize species of service, hence more than one possible taxonomy. Initially, alternative taxonomies will provide rather different factorings of behavior. In that case, services will have to publish/register themselves in different places in the different hierarchies. Over time, we can expect convergence of the more common services. However, we also can expect constant emergence of new service categories and retirement of outmoded services, which will always generate a need for minor (and sometimes major) reorganization of the taxonomic categorizations.
- Taxonomic categories must be human engineered like Yahoo's or Netscape's Open Directory to reflect the semantics of the categorized services. Each category must define the semantics (what the service "does") as well as the API (how one invokes the service). We propose that a category be defined by a *Canonical Service Description* (CSD). All services in the category must implement the canonical service description (although they can do more or do it "better/faster/cheaper"). These service descriptions must include human readable as well as machine parsable information. And they must also describe non-functional requirements such as security and authentication assumptions, and other prerequisites considered to be common to all services in that category.
- The canonical service description for each category should be related to that of the category one level up. But the semantics of a given service may have to differ from the next level up in important ways. That is, a specialization of a service at the next level down in the hierarchy may implement the same verb, but do so in a very different way with different argument characteristics and different returned results. So the canonical service description should describe how the services in the category differ from those in their parent category and from any sibling categories in the hierarchy.
- Since canonical Service Descriptions are service descriptions that *define* a taxonomic category, they must provide executable tests, described in XML, for membership in the category, i.e., a test for each "method" in the service, which can be run against any service proposed as a candidate for a given category. These tests allow the broker to "objectively" and automatically determine that a service satisfies the requirements for being included in that category. It should be noted that services can spoof these tests by artificially ensuring that the canonical tests work. Running the tests is simply a mechanism for doing a first cut automated membership test. Social and legal constraints will also evolve to discourage spoofing. And, sophisticated tests can be run by crawlers that will be more difficult to spoof. Security constraints on some services may require that the broker be given special certificates or other authentication information as a part of registration so that it can execute the tests. In situations that call for extreme security, authentication, or encryption, the broker may only receive exceptions in response to the tests. Even in that case, though, it should receive the correct exceptions, i.e., the exception received should indicate that the service is present but not responding due to inadequate authentication. The conflict between testability and security is not likely to be severe with general purpose brokers. Highly secure services are likely to be registered with special purpose brokers that themselves are secure and have special purpose mechanisms for testing.
- Services may be registered in multiple categories, although in that case the service must be able to function correctly in each of those categories.

Service Descriptions

Service descriptions play a central role in publishing services for use, categorizing them, and finding the right service to satisfy a need. In other words, they are the medium of information exchange that supports publish, find, and bind. And their role as canonical category descriptions is the *key* element of transforming human readable taxonomies, such as present day Yahoo categories, into the machine-readable taxonomies required for automated B2B e-services. A companion document will explain the format and content of Service Descriptions in detail [see The Service-Oriented Architecture Overview]. It is sufficient here to note the high-level requirements.

- Service descriptions must be implemented in XML. They need to be both human readable and machine parsable. They will also have to be extensible since the evolution of e-business services cannot be predicted with any confidence.
- Service descriptions must provide all of the semantic information needed for requester of the service to decide whether it satisfies their requirements and for the broker to decide on categorization.
- Service descriptions must describe APIs in an automatically testable manner. If the service implements exactly the API specified in the category's canonical service description, the service description need not provide any further tests. However any additional services must be accompanied by additional tests, and services may wish to compete on robustness by providing more detailed tests that demonstrate competitive advantage.

Service descriptions must also specify non-functional requirements. The two most important issues are security/authentication/privacy issues related to the exchange of information necessary for the service to be consummated, and legal/contractual issues between the provider and requester of the service (issues such as those dealt with in Trading Partner Agreements which will be incorporated into or pointed at by service descriptions).

Mutation of categories and service descriptions

Since the marketplace of services will be in constant flux, there will be occasions when brokers find it necessary to create and destroy categories, or change the canonical service description for a category. It will be the broker's responsibility to notify interested parties of these changes and provide appropriate version management of categories. Techniques for change and version management will evolve. One approach is to mark categories as "obsolete" as a sign that users of those categories should change to the new categories in a timely manner. The broker must also notify the service providers of the change. Service providers will need to maintain their services, and service descriptions in the face of categorization changes. Of course, they may simply leave their services unchanged and be content to stay with other categorization services or to stay in a category marked obsolete.

The requester will also need to guard against the possibility that categories, and even service descriptions, may change. The requester might save a copy of the desired canonical service description. That copy can be checked against the actual service description at runtime, or crawlers can check periodically for changes and notify the maintainers of the systems when changes are found. Similarly, discovery at runtime that a category has been market as "obsolete" should trigger a message to system designers and maintainers so they can update their design.

Principles of collaboration at run time

Run time is when the carefully scripted and rehearsed sequence of collaborations between the various services plays out. Service requesters must find and bind to the right instance of one or more other services and then carry on the dialog needed to get the job done. Those services may, in turn, need to take advantage of still other services on other servers. The collaboration itself, not the behavior of any one machine, provides the value.

For a decade or so, the consensus notion of computing has been undergoing a transformation from "computing happens in a single CPU" to "the network *is* the computer." We have seen architectures based on distributed object systems communicating via remote method calls and marshaled/demarshaled objects, e.g., CORBA. This approach stems from a desire on the part of architects to tame the complexities of collaborating computers by making them look more like a problem with which we are more familiar: object systems running in a single computer. These systems provide ways to specify the APIs supported by the various remote objects, but not the semantics. Such systems succeed in relatively small scale applications where shared assumptions can be enforced by the system designers. Collaborating services in the vast untamed chaos of the Internet is another beast entirely.

As we think about appropriate communication architectures for service-oriented computing, we should study analogous systems of similar communication complexity. Social and economic systems have the requisite complexity, but they are lubricated by human judgment. We need an analogy that does not rely on that human input if we wish to apply it to automated B2B interactions. Biological multi-cellular communication strategies provide the best parallel. In fact, many aspects of the transition from single-CPU computing to network computing seems to be paralleling the transition from single-cell to multi-cell organisms.

One of the first problems multi-cellular organisms had to solve was how the cells communicate and collaborate effectively. Multi-cellular communication strategies have evolved by trial-and-error for more than a billion years. We cannot know how many communication mechanisms have been tried and found wanting during that evolution, nor what the principles of communication in these failed mechanisms might have been. But we can look at the surviving successful techniques for useful analogies.

Present day cells in multi-cellular organisms communicate with each other primarily via molecular messages, either by broadcasting these messenger molecules (e.g., through the circulatory system), or by passing the messenger molecules directly to adjacent cells [See "The Touchstone of Life, Werner Loewenstein, Oxford University Press, New York, 1999]. Even neurons, which transmit messages along their length by electrical impulses, communicate their message to the next neuron in the neural circuit by sending neurotransmitter molecules across a narrow intercellular gap called a synapse.

Molecules as tiny as nitric oxide can act as messengers. However, complex and selective messages require complex messenger molecules that are constructed of long chains of subunits. Evolution has settled on two very different sorts of chain molecules: proteins, which are chains of amino acids, and DNA or RNA which are chains of nucleotides. Both sorts are complex enough to carry sufficient information. However, they play very different biological roles because of their relationship to the cellular CPU. Proteins make up the various structural and dynamic parts of the cellular machine whereas DNA or RNA genetic sequences provide the fundamental programming code that directs each cell's machinery. Thus the transfer of genetic material, in effect, reprograms the receiving cell. The transfer of all other sorts of messenger molecules act, instead, as polymorphic messages in the sense that the receiving cell, not the sender, determines what response is appropriate.

In single cell organisms such as bacteria, directly transferring genetic material in the form of DNA or RNA from cell to cell is a normal and powerful means of information transfer. It allows successful mutations to quickly spread to a large population of bacteria. For example, direct DNA transfer is partly responsible for the rapid spread of antibiotic resistance among bacteria species. However, healthy multi-cellular organisms transmit DNA only in the very special circumstance of sexual reproduction. Viruses, which carry genetic material from one cell to another, violate that rule to the detriment of the organism infected by the virus. The rule against genetic transfer is so universally obeyed in multi-cellular organisms that Loewenstein [p. 277] calls it "…the taboo of intercellular transfer of genetic information."

The evolution from single CPU computing to multi-machine computing has taken place in a heterogeneous computing environment. As a result, we have developed universal data (in the form of XML) and universal code (Java). In principle, either could be the basis for B2B communication. However, we should consider the lessons from multi-cellular evolution as a strong vote for polymorphic XML data messages and against messages comprised of executable code such as Java or ActiveX. Experience with computer viruses on the Web is leading B2B pioneers to a similar conclusion. Many government and commercial organizations, where security is a sensitive issue, have already developed a taboo against the automated transfer of executable code between machines. Most of the B2B initiatives assume XML as the format for variable portions of messages, e.g., data needed by the requested service and data returned from the service.

Security and authentication

B2B services will often require that sensitive information be sent between machines. And, many services will require a trusted way to ensure that the requester is known and authorized. So B2B services will require a message architecture that:

- keeps information secure when needed (encryption),
- provides a trusted way to identify the sender and receiver of a message (authentication), and
- ensures that only authorized messages are acted upon (authorization).

A separate document on the security issues related to service-oriented architectures will be forthcoming ["Security and Trust" by Maryann Hondo].

Exception Handling

Especially in an Internet-wide service net, things are going to go wrong. Where messaging is very efficient, i.e., in a single system or within the firewall on a fast LAN, a call and error return scheme may work well enough. But when services may take long and unpredictable times to return, simply waiting for an error return is unsatisfactory. In that case, we will need some better exception event model. That will require a taxonomy of exceptions that is largely orthogonal to the taxonomy of services. Service descriptions will need to specify their exceptions in their API interface (perhaps in a manner similar to Java).

The business of e-business services

Providing services

Providing e-business services is the business of exposing your internal IT environment to the outside world in order to bring your customers, suppliers and partners into a closer relationship with your business. Each separate business must decide which services to expose, how to make

tradeoffs between security and easy availability, how to price the services (or, if they are free, how to exploit them for other value). Each company will also have to decide what category the service should be listed in for a given broker service and what sort of Trading Partner Agreements are required to use the service.

Services can be wrappered legacy function or new software intended to be a service. However new software may also be little more than scripted collaborations of other services provided internally or externally. That is, we expect that as an economy of services emerges, opportunities for new services will appear that require little more than the orchestration of, and delegation to, several existing services.

All service providers will share the desire to get a service up and running quickly once the decision is made. Tools for quickly building and/or wrappering services will be strategically important in the B2B arms race. Tools may also be needed to help service providers to keep appropriate usage counts and perhaps to mine the usage data. Micropayment or other e-commerce tools may also be needed. Business opportunities for providing tooling and middleware will be plentiful.

Companies will also share the desire to quickly get new services listed on one or more brokers and to advertise/market them to the right audience. Business opportunities are likely to arise to provide services to help with these shared issues.

Using services

The business reasons for using e-services will be as varied as the reasons for using IT in general. The "killer" application for e-business services today is supply chain integration. Others will emerge as well. What we expect to happen is that a marketplace of e-services will create opportunities for outsourcing some work that is now done in monolithic applications. Another obvious application is splitting monolithic internal applications into multiple services that can more easily be specialized. This sort of thing is already common in high volume web sites.

One important issue for users of services is the degree to which services are statically chosen by designers versus dynamically chosen at run time. Since dynamic service choice is the eventual goal, the architecture must support a fully dynamic ecology of services. Even if most initial usage may well be largely static, any dynamic choice opens up the issues of how to choose the best service provider, and how to assess quality of service. Another issue is how the user of services can assess the risk of exposure to failures of service suppliers. And what techniques will evolve to reduce those risks. We propose below the notion of e-business service crawlers as mechanisms to help answer these sorts of questions.

Brokering services

Service brokers provide the service of creating and managing taxonomies, evaluating and registering services, and providing rapid lookup of that information to interested parties.

The e-business services economy will offer more than one sort of broker. Some brokers will specialize in breadth of listings. Others will offer high levels of trust in the listed services. Some will cover a broad landscape of services and others will focus within a given industry. Undoubtedly brokers will also arise that simply catalog other brokers. Depending on the business model, a broker may attempt to maximize lookup requests, number of listings, or accuracy of the listings. Brokers will also compete on the merits of their taxonomies, i.e., having the "best factored" hierarchy to maximize ease of finding desired services.

To use a term from modern "Webspeak" one key issue for brokers will be how to "monetize" their services. It is unlikely that brokers will be able to charge for basic brokering services. At least in the initial stages of the evolution of brokering, there are unlikely to be substantial barriers to entry. Setting up brokering services will be easy enough that many competitors will arise. Among these, many will mimic the Yahoo model by providing look-up service at no charge. Brokers that provide extra levels of service may be able to charge for the information. Extras might include Dun & Bradstreet sorts of assurances, e.g., credit checking, "rating" services, customer feedback gathering (on both sides of the service transaction), and the like. However, the history of the Web over the last five years is that most attempts to extract "per click" fees are shunned. Moreover, at least for those brokering services that offer breadth, there is a network effect (i.e., a winner-take-all dynamic). The larger the listing, the more service requesters will use the broker, therefore the more value there is in having a new service listed with that broker. This dynamic will drive initial entrants to push for "market share" at the expense of revenues. Charging for brokering services in that kind of market will be suicidal.

However, as a byproduct of their service in matching providers and requesters, brokers will be able to gather data on supply and demand for services. This data about service usage may have market value in and of itself. Brokers are uniquely positioned to obtain marketing data that can be sold to potential service providers and requesters. Brokers know the kinds of services looked for, and the numbers of customers looking for each service. Traffic by category could also provide estimates of the value of specialization versus generalization. Brokers may also be in a position to estimate requester's tradeoffs between service cost and QOS. Brokers could mine historical data from their listings to estimate other characteristics such as the diversity of successful servers/services and whether services are winner-take-all categories or commodity categories. These sorts of information can be made available for a fee or as part of a consulting service. Certainly, Yahoo has shown creativity in extracting value from categorization services that are nominally free. Pre-internet analogs such as the Yellow Pages also uncovered novel business advantages unforeseen by the original founders of the service.

Co-evolution of the categories and the services fulfilling them

The design-time issue for service requesters is that the requester must choose the types of services and the characteristics of the specific services to be later found and bound. The design issue for brokers is how to choose and organize the taxonomic categories in which to place services. Brokers must also ensure the service belongs in a given category, and perhaps rank services in the same category. The issue for service providers is what services to provide with what API's, and how to get them registered in taxonomies. For each of the three roles, choices of the best techniques for dealing with their respective issues depend in part upon choices made by the other parties. Each player will improve its techniques, which will change the landscape so that other players will need to respond. Thus techniques, taxonomies, and services will co-evolve over time.

Crawling and monitoring the web of services

Services will be provided by organizations with many different agendas, standards of quality and organizational lifetimes. The quality of service and even the existence of services may change unpredictably. Thus, there will be a need to constantly evaluate the accuracy of taxonomies. We propose that this job be done by "taxonomy crawlers." These are servers that continuously crawl taxonomies testing for presence of listed services (i.e., that the service is still there), and for response times, whenever possible. Results will constantly update the crawler's QOS database and are available as a separate service (perhaps analogous to a Dun & Bradstreet for services). A given taxonomy service should have its own QOS crawler to rate or rank services within a

category. But external crawlers will be required to provide a more objective service since the taxonomy service crawler may be biased (e.g., by pressure from service providers to "make them look good"). In effect, crawlers enforce taxonomies while also providing pseudo real-time QOS data to help choose between otherwise equivalent services.

Note that security, authentication, and the like may frustrate independent crawlers because they will not be able to access protected services. However, brokers can require that registering services provide appropriate "guest" certificates or passwords for use by their crawlers.

The existence of possibly many competing brokers with different taxonomies will also create the need for what might be called meta-crawlers, i.e., taxonomy matchers that crawl competing taxonomies, attempting to match common points and thereby find services similar to a given service. For example, a meta-crawler might find all competing taxonomies (or portions of the same taxonomy, since a service might be listed in more than one place in a hierarchy) that contain the same service or whose canonical service descriptions are compatible (i.e., supersets). The assumption, which could be tested via the executable API tests, is that services in such categories are also useable. Super and sub categories can also be examined on the theory that they should also be "similar." Note that crawling taxonomic categories is a much more efficient way to find similar services than brute force checking of all possible services for compatible APIs. However, once a meta-crawler has found "similar" categories, it can afford the effort to do brute force matching of the services it finds in those categories.

Broker monopolies?

It should be noted that the tendency of broker markets to be winner-take-all, together with the operation of meta-crawlers, may tend to quickly force convergence of competing brokers and their taxonomies toward one winning broker in a given market. If taxonomies are freely copyable, this will not give monopoly power to the winning broker. However, this issue has the potential to be snarled in Intellectual Property (IP) Law. There have been decisions on just what IP is owned by telephone Yellow Pages and white pages that will bear on taxonomy IP.

Neither service providers nor service requesters will want to be hostage to any one broker. So it is in the interests of most parties to ensure, if at all possible, that brokers provide open taxonomies, i.e., that they compete on speed, quality, and timeliness, or other auxiliary services, but that they do not own the taxonomy itself. Encouragement of open-source copyright licenses modeled after the GPL (i.e., viral licenses) might achieve this goal. If that approach does not prevail service requesters might initiate a practice requesting services from at least two brokers, i.e., deliberately rejecting single source brokering. Since service requesters would receive no obvious benefit from monopolistic brokers, they should be willing to cooperate with such a convention.

Conclusions

What we have discussed here is an ecology composed of service providers, service requesters, service brokers, taxonomies, service crawlers, meta-crawlers, the businesses that weave services into their operations, and of course the businesses that provide software, hardware, and services for the development and maintenance of the ecology. As with all economic ecologies, once they mature they are robust and provide many niches for profitable exploitation. However, they are quite fragile in the early stages.

While the ecology may well form without any explicit efforts, it is in the best interest of those companies that stand to profit from the mature ecology to make every effort to help bring it

about. And IBM cannot do it alone. We should involve other important players where possible, to help "jump start" and nurture the ecology in its early stages.

It is important that all of the companies that attempt to jump start this ecology see the long term value of their efforts as more important than any short term value to be gained by attempting to bias the ecology for short term interest. Premature attempts to exploit the ecology for profit will most likely leave it stillborn. In particular, all players should avoid attempts to exclusively "own" key intellectual property such as the initial taxonomy, rights to formats such as service descriptions, or to ideas such as crawlers, meta-crawlers, or business processes necessary to the ecology. Attempts of that sort will most likely frighten off the businesses that otherwise would provide the services or base their business on using the services provided.