E-Services: Current Technology and Open Issues

Thomi Pilioura* and Aphrodite Tsalgatidou†

University of Athens
Department of Informatics & Telecommunications
TYPA Buildings, Panepistimiopolis, Ilisia
GR-157 84, Athens, Greece
{thomi,afrodite}@di.uoa.gr

Abstract. The Internet changes the way business is conducted. It provides an affordable and easy way to link companies with their incorporating trading and distribution partners as well as customers. However, the Internet's potential is jeopardized by the rising digital anarchy: closed markets that cannot use each other's services; incompatible applications and frameworks that cannot interoperate or build upon each other; difficulties in exchanging business data; lack of highly available servers and secure communication. One solution to these problems is a new paradigm for e-business in which a rich array of modular electronic services (called e-services) is accessible by virtually anyone and any device. This new paradigm is currently the focus of the efforts of many researchers and software vendors. This paper presents the e-services architecture, its advantages as opposed to today's applications and gives an overview of evolving standards. It then presents the related technical challenges, the way some of them are addressed by existing technology and the remaining open issues.

1 Introduction

Companies face a number of challenges in choosing and implementing the right software and technology solutions in order to better serve their needs and support their business endeavors. This has become particularly problematic in recent years as companies attempt to leverage existing practices, systems and resources across the Web. Critical to success in this environment is to find an integrated, robust e-business solution that allows a company to leverage existing applications, rapidly adapt to the unique needs of the business, and continually evolve as business requirements change over time.

Nowadays, the current trend in the application space is moving away from tightly coupled systems (e.g. DCOM based business solutions [1]) and towards systems of loosely coupled, dynamically bound components (e.g. Jini [2] or Enterprise Java

* Ms. Thomi Pilioura is also with the National Bank of Greece, email: tpilioura@nbg.gr.
† Corresponding Author.

F. Casati, D. Georgakopoulos, M.-C. Shan (Eds.): TES 2001, LNCS 2193, pp. 1-15, 2001.

Beans [3]). Systems built with these principles are more likely to dominate the next generation of e-business systems, with flexibility being the overriding characteristic of their success.

A new paradigm for e-business [4, 5], called e-services, seems to be able to help in this direction. E-services are self-contained, modular applications that can be described, published, located and invoked over a network. The e-services framework enables an application developer who has a specific need to cover it by using an appropriate e-service published on the Web, rather than developing the related code from scratch.

The e-service architecture is the logical evolution from object-oriented systems to systems of services. As in object-oriented systems, some of the fundamental concepts in e-services are encapsulation, message passing and dynamic binding. The e-service approach can be considered as a component-based approach where components are large and loosely coupled. Furthermore, the e-service approach advances the component-based paradigm a step beyond signatures, since information related to the quality of service and to what it does is also published in the service interface.

The goal of this paper is to give a technology overview of the challenging area of e-services and to draw some conclusions regarding the status, the applicability and the future of e-service technology. It is therefore organized as follows. Section 2 presents the e-services architecture and its anticipated benefits as compared to today's applications. Section 3 gives an overview of evolving standards. Section 4 presents the technical challenges deriving from the e-services architecture, the way some of them are addressed by current technology and the remaining open issues. Finally, we present our concluding remarks.

2 The E-Services Concept

E-services constitute a new model for using the Web. It allows the publishing of business functions to the Web and enables universal access to these functions. The architecture that enables it, is presented in the following paragraphs along with the benefits that this architecture could bring to e-business.

2.1 The E-Services Architecture

Several basic activities need to be supported by any service-oriented environment:

1. An e-service needs to be created and described.
2. An e-service needs to be published to one or more Intranet or Internet repositories for potential users to locate.
3. An e-service needs to be located by potential users.
4. An e-service must be invoked to be of any benefit.
5. An e-service may need to be unpublished when it is no longer available or needed, or may need to be updated to satisfy new requirements.

In addition to these basic activities there are some other activities that need to take place in order to take full advantage of the e-services architecture. Such activities include e-services integration with existing infrastructure, e-services management and

maintenance. However, we consider that the e-services architecture requires at least the following basic operations: describe, publish, unpublish, update, discover and invoke, and contains 3 roles: service provider, service requester and service broker [6]. Fig. 1illustrates the e-services architecture.

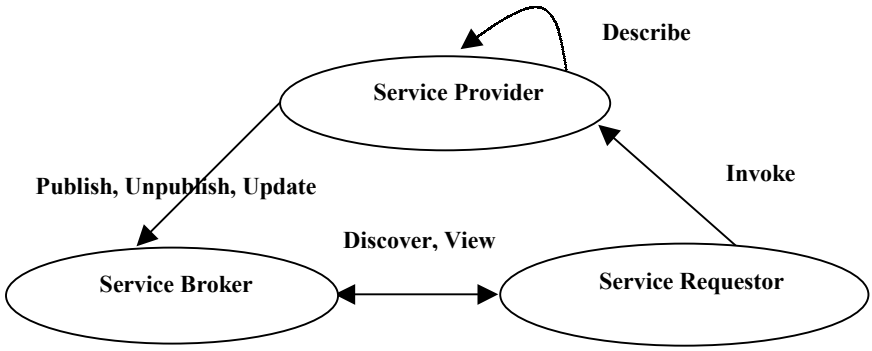

Fig. 1. E-Services Architecture.

Service Provider: A service provider is the party that provides software applications for specific needs as services. Service providers publish, unpublish and update their services so that they are available on the Internet. From a business perspective, this is the owner of the service. From an architectural perspective this is the platform that holds the implementation of the service.

Service Requestor: A requestor is the party that has a need that can be fulfilled by a service available on the Internet. From a business perspective, this is the business that requires certain function to be fulfilled. From an architectural perspective, this is the application that is looking for and invoking a service. A requestor could be a human user accessing the service through a desktop or a wireless browser; it could be an application program; or it could be another e-service. The requestor finds the required services via the Service Broker and binds to services via the Service Provider.

Service Broker: This party provides a searchable repository of service descriptions where service providers publish their services and service requesters find services and obtain binding information for services. It is like telephone yellow pages. Such examples of service brokers are the e-speak E-services Village [7], UDDI (Universal Description, Discovery, Integration) [8], [9], [10] and XMethods [11].

2.2 Anticipated Advantages of E-Services

The following paragraphs focus on the anticipated benefits of e-service approach as compared to today's applications.

Interoperability: Any e-service can interact with any other e-service. This is achieved through an XML-based interface definition language and a protocol of

communication. By limiting what is absolutely required for interoperability, interacting e-services can be truly platform and language independent. This means that developers should not be expected to change their development environments in order to produce or consume e-services. Furthermore by allowing legacy applications to be exposed as services, the e-services architecture easily enables interoperability between legacy applications or between e-services and legacy applications.

Just-in-Time Integration: Traditional system architectures incorporate relatively brittle coupling between various components in the system. These systems are sensitive to change. A change in the output of one of the subsystems or a new implementation of a subsystem will often cause old, statically bound collaborations to break down. E-services systems promote significant decoupling and just-in-time integration of new applications and services, as they are based on the notion of building applications by discovering and orchestrating network-available services. This in turn yields systems that are self-configuring, adaptive and robust with fewer single points of failure.

Easy and Fast Deployment: Enterprises using the e-service model are expected to provide new services and products without the investment and delays a traditional enterprise requires. They may utilize the best-in-their-class component services without having to develop them (outsourcing).

Efficient Application Development: Application development is also more efficient because existing e-services can be reused and composed to create new e-services. An e-service can aggregate other e-services to provide a higher-level set of features.

Strong Encapsulation: All components are services. What is important is the type of behavior a service provides, not how it is implemented. This reduces system complexity, as application designers do not have to worry about implementation details of the services they are invoking.

3 Evolving Standards for E-Services

Today there is a lot of activity in the e-services area. We are currently witnessing the rapid development and maturation of a stack of interrelated standards that are defining the e-services infrastructure. In this section we will outline the key standards of WSDL (Web Services Description Language) [12], SOAP (Simple Object Access Protocol) [13] and UDDI (Universal Description, Discovery, Integration) [14, 15]. Each one of these standards supports some of the basic operations of a service-oriented environment, namely describe, publish, unpublish, update, discover and invoke.

3.1 Web Services Description Language (WSDL)

For an application to use an e-service, the programmatic interface of the e-service must be precisely described. In this sense, WSDL plays a role analogous to Interface Definition Language (IDL) used in distributed programming. It is an XML grammar for specifying properties of an e-service such as *what* it does, *where* it is located and *how* to invoke it.

A WSDL document defines *services* as collections of network endpoints, or *ports*. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: *messages*, which are abstract descriptions of the data being exchanged, and *port types* that are abstract collections of *operations*. The concrete protocol and data format specifications for a particular port type constitute a reusable *binding*. A port is defined by associating a network address with a reusable binding, and a collection of ports defines a service. Hence, a WSDL document uses the following seven elements in the definition of network services:

- *Types:* A container for data type definitions using some type system (such as XSD).
- *Message:* An abstract, typed definition of the data being communicated.
- *Operation:* An abstract description of an action supported by the service.
- *Port Type:* An abstract set of operations supported by one or more endpoints.
- *Binding:* A concrete protocol and data format specification for a particular port type.
- *Port:* A single endpoint defined as a combination of a binding and a network address.
- *Service:* A collection of related endpoints.

3.2 Simple Object Access Protocol (SOAP)

SOAP [13] is a standard for sending and receiving messages over the Internet. It is independent of the programming language, object model, operating system and platform. It uses HTTP as the transport protocol and XML for data encoding. However, other transport protocols may also be used. For example, Simple Mail Transport Protocol (SMTP) can be used to send SOAP messages to e-mail servers.

SOAP defines two types of messages, Request and Response, to allow service requestors to request a remote procedure and to allow service providers to respond to such requests. A SOAP message consists of two parts, a header and the XML payload. The header differs between transport layers, but the XML payload remains the same. The XML part of the SOAP request consists of three main portions:

- The *Envelope* defines the various namespaces that are used by the rest of the SOAP message.
- The *Header* is an optional element for carrying auxiliary information for authentication, transactions, and payments. Any element in a SOAP processing chain can add or delete items from the Header; elements can also choose to ignore items if they are unknown. If a Header is present, it must be the first child of the Envelope.

– The *Body* is the main payload of the message. When SOAP is used to perform an RPC call, the Body contains a single element that contains the method name, arguments, and e-service target address. If a Header is present, the Body must be its immediate sibling; otherwise it must be the first child of the Envelope.

A SOAP response is returned as an XML document within a standard HTTP reply. The XML document is structured just like the request except that the Body contains the encoded method result. The XSI/XSD tagging scheme is optionally used to denote the type of the result.

3.3 Universal Description, Discovery, Integration (UDDI)

UDDI defines a common means to publish information about businesses and services. It can be used at a business level to check whether a given partner offers a particular e-service, to find companies in a given industry with a given type of service, and to locate information about how a partner or intended partner has exposed an e-service in order to learn the technical details required to interact with that service.

The UDDI specifications consist of an XML schema for SOAP messages, and a description of the UDDI APIs specification.

The UDDI XML schema defines four key data structures: business entities, business services, binding templates and tModels. Business entities describe information about a business, including their name, description, services offered and contact information. Business services provide more detail on each service being offered. Each service can then have multiple binding templates, each describing a technical entry point for a service (e.g., mailto, http, ftp, etc.). Finally, tModels describe what particular specifications or standards a service uses. With this information, a business can locate other services that are compatible with its own systems. UDDI also provides identifiers and categories to mark each entity using various taxonomies (related industry, products or services offered and geographical region).

A UDDI Business Registry is itself a SOAP e-Service. IBM, Microsoft, and Ariba are implementing UDDI Registries. Service providers will only have to register at one of the implementations since updates to any are replicated in the others on a daily basis. Each of the business registries provides operations to create, modify, delete, and query each of the four data structures. All these operations can be performed either via a Web site or by using tools that make use of the UDDI API specification.

The UDDI APIs contain messages for interacting with UDDI registries. Inquiry APIs are provided to locate businesses, services, bindings, or tModels. Publishing APIs are included for creating and deleting UDDI data in the registry. The UDDI APIs are based on SOAP.

4 Technical Challenges: Current Technology and Open Issues

For e-services to succeed, there are a number of technical challenges that have to be met, many of which are related to the open and hostile environment in which they have to survive. These challenges are related to:

1. The e-services life cycle, which includes activities such as the description, publishing, discovery, invocation, integration and management of an e-service.
2. Some initial activities that have to be performed by the users before searching for a suitable e-service, e.g. requirements analysis and description of their needs.

The challenges can be split into categories accordingly to three different dimensions: the complexity of the e-services, the indispensability of the operations performed on e-services and the level of the abstraction related to the operations, see Fig. 2.

Services may be simple or composite. Composite services consist of a number of simple services. Operations to simple or composite services may have basic or value-added functionality.

- Basic operations concern functionality necessary to be supported by every e-service environment, namely e-service description, publishing, discovery and invocation.
- Value-added operations bring value-added functionality and better performance to the e-service environment, e.g. e-service management, security and accountability.

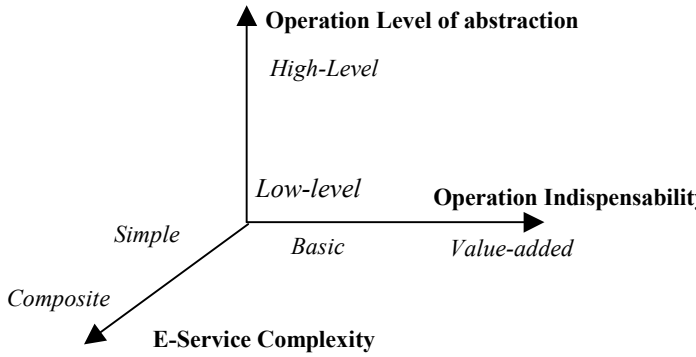

Fig. 2. Dimensions of Technical Challenges for E-Services.

All kinds of operations, either basic or value-added applied either to simple or to composite services are expected to expose their functionality at two different levels:

- At a lower abstraction level, i.e. at a level mainly concerned with the syntactic or implementation aspects of an e-service
- At a higher abstraction level, i.e. at a level where the main focus is on the semantic or conceptual aspects of e-services.

Low level operations are tailored towards the programmers' requirements, while high level operations facilitate the work of business users by shielding off the lower level technical details. It is expected for the e-service paradigm to prevail if all desirable operations are offered at a high abstraction level.

The above three-dimensional categorization results in the eight sub-cubes depicted in Figure 3. Each sub-cube describes a specific category of technical challenges:

1. Techical challenges related to low level (syntactic level) basic operations to simple e-services.

2. Techical challenges related to high level (semantic level) basic operations to simple e-services.
3. Techical challenges related to low level value-added operations to simple e-services.
4. Techical challenges related to high level value-added operations to simple e-services.
5. Techical challenges related to low level basic operations to composite e-services.
6. Techical challenges related to high level basic operations to composite e-services.
7. Techical challenges related to low level value-added operations to composite e-services.
8. Techical challenges related to high level value-added operations to composite e-services.

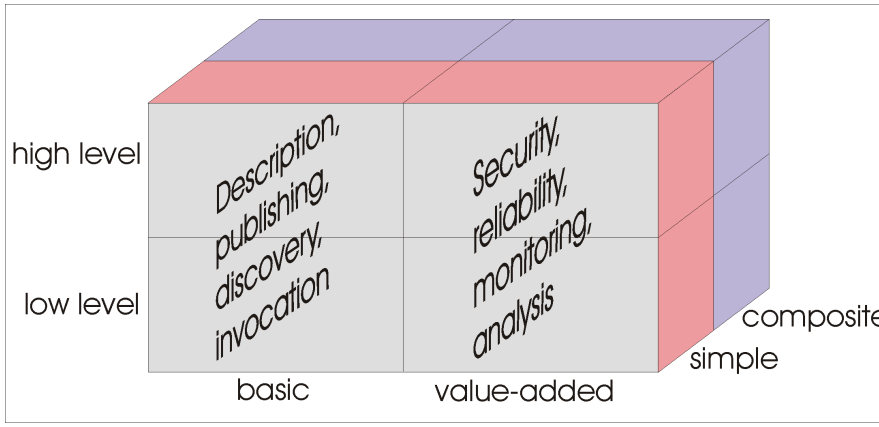

Fig. 3. Cubiform Representation of Technical Challenges for E-Services.

The standards presented in the previous section mainly address the first category of technical challenges, i.e. they provide basic operations to simple services at a low abstraction level, for example the format of messages needed to invoke an e-service.

There are however some other initiatives, research projects and vendor products [16, 17] that offer interesting solutions to the other categories of challenges. In the following paragraphs we examine the challenges for basic and value added operations and how some research projects and software products address them. The first subsection describes the technical challenges for low level or high level *basic* operations performed on simple or composite services. The second subsection describes the technical challenges for low level or high level *value-added* operations performed on simple or composite services.

4.1 Technical Challenges for Basic Operations

Description: The description of an e-service can be at both syntactic and semantic level. The WSDL, analyzed in section 3, describes the syntactic aspects of a service.

Essentially, a WSDL document describes how to invoke a service, and provides information on the data being exchanged, the sequence of messages for an operation and the location of the service. WSDL is not able to sufficiently capture the semantics of the business domain and the structure of the service (e.g. the sub-services, the part of the services).

The Advertisement and Discovery of Services (ADS) protocol specification [19] supports the following three classes of service descriptions: the description of a simple service, the description of a collection of services, the description of a repository of services. ADS uses the service description architecture of WSDL. Therefore, it describes the syntactic aspects of a service.

The Service Description Language (SDL) described in [18], addresses e-service description at both lower and higher level and offers constructs to describe:

- Service properties i.e. service general information (e.g. identification, owner), service access information (e.g. service location-URL, public key certificate) and contact information
- Service ontology i.e. what is the service about and the terminology that is needed to discover the service.
- Service cost i.e. the estimated cost for using the service or the information provided by the service
- Payment i.e. the way the service receives the payment from the customers
- Actors, i.e. the people or organizations using the service. However, SDL does not specify how this is possible if e-services are enacted through a web interface.
- Authorization/security/visibility, i.e. who can see/use what (service contents and functions).
- Service contents: which specifies the content and the structure of the underlying service e.g. the attributes, objects, constraints on use of attributes/objects etc. For example a car information object can have attributes such as brand, model, color and mileage. SDL specifies the data types of the attributes to guarantee type safe orchestration of e-services. However, this violates encapsulation.
- Service capability: which specifies the service structure/components, the conditions of using the service and the order of component invocation.

SDL documents on one hand need to be easily understood by non-technical end-users without the technical details such as network and security information. On the other hand, programmers who are responsible for e-service integration require these last categories of information. Thus they have defined various views of an SDL document: the yellow page view, the white page view and the technical green page view. The first two views particularly aim to inform business users, whereas the last view is tailored towards the requirements of the implementers.

In ebXML [20], the CPP (Collaboration Protocol Profile) [21] describes the service provider, the role it plays, the services it offers and the technical details on how these services may be accessed.

In e-speak [22], e-service providers provide a description of the e-service that consists of attributes specific to that service. The more attributes registered for e-services, the richer users' requests for services can be. Technically speaking, the attributes are meta-data expressed in XML. They have to conform to a specific agreed-upon vocabulary developed for that type of service. Service providers may

create their own vocabularies or select a Microsoft BizTalk vocabulary, or use a combination of the two.

Publishing: Publishing is one of the basic operations as it makes a service known and available to be used. Publishing just like description can be at both syntactic and semantic level. The offered solutions include the ones provided by UDDI, e-speak E-services Village, eFlow [23] and ebXML Registry and Repository [24]. E-services Village is an online community for e-Speak developers to register their services. eFlow provides a repository of processes, service nodes and service data type definitions. The repository is organized into a hierarchy dividing the former into groups and simplifying its browsing. The ebXML Registry has schema documents, business process documents and CPP documents. Requestors communicate with the Registry using the ebXML Messaging Service [25].

Given the existence of these registries, the Advertisement and Discovery of Services Protocol (ADS) mentioned above, asks the question "how do I facilitate the building of a crawler that can pull e-services advertisements off the Web, without people having to push advertisements for their services to the registry?" ADS supports two mechanisms that allow service providers to announce the availability of service descriptions: the creation of a file placed in the root directory of their server and the use of a meta tag which can be included in any HTML file to denote the location of the service description.

Discovery: Discovery of services is a basic operation that has to be supported at a high level, as it is important to be able to describe your needs at a conceptual level and to make sure that your needs are matched with what the service provides. Both WSDL and UDDI address this issue only at a low-level. The Service Request Language (SRL) proposed in [26] attempts to tackle this issue at a higher level. SRL is a declarative language for querying and requesting an e-service described using SDL. This language is able to query the system at two levels: metadata level (service location and semantic exploration) and service level (connect to the actual services and use their operations). E-speak also addresses the discovery issue at a conceptual level. In e-speak, users just have to express their service request as a collection of attribute values. Then, e-Speak will automatically discover registered services that have the desired attributes.

Invocation: This operation is mainly addressed at the syntactic level by SOAP and the emerging W3C XML Protocol (XP) [27]. Invocation is also addressed by the ebXML Message Service, which, in contrast to SOAP, provides a Quality of Service (QoS) framework that ensures reliable message delivery. The QoS framework can be extended to support security and transaction semantics.

4.2 Technical Challenges for Value-Added Operations

Composition: Composition refers to the combination of basic e-services (possibly offered by different companies) into value-added services that satisfy user needs. An interesting issue here is to find which architectures, models, and languages can achieve zero latency e-service composition. eFlow [28, 29] supports the dynamic

composition of simple e-services at a semantic level. It allows configurable composite services, where different customers select the desired subset of features that this composite service can offer. It also provides dynamic service process modification (ad-hoc changes and dynamic process evolution), to adapt to changes in the business environment. Another solution is provided by the Polymorphic Process Model (PMM) [30] which also enables dynamic composition of e-services at a semantic level by separating the service interfaces from service implementations and defining the service interfaces in terms of service state machines and its input/output parameters. A further advantage of this is the ability to use multiple implementations for the same service without the need to modify the specification of the running process that invokes the service.

Integration: Integration means associating a new e-service with another e-service or other resources such as databases, files, directories and legacy applications. The need of integrating several e-services to perform a business function generates a lot of issues, ranging from understanding and accessing the different APIs with the appropriate communication protocol, to managing the different security requirements, data formats, programming models and transmission models. To further complicate the picture, different e-services often run on top of heterogeneous hardware and software platforms. What we need is easy integration, namely use of e-services without having to write integration code before. A solution to this problem is provided by Enterprise Integration (EAI) platforms. Integrators such as Tibco's ActiveEnterprise and ActiveExchange [31], SeeBeyond's eGate [32] or BEA's eLink [33], WebMethods' solutions [34], ease the daunting task of connecting different applications, by providing uniform access to heterogeneous systems and by offering design tools that help system integrators in appropriately accessing the different applications and managing their results. If fundamental re-implementation of business operational systems is required, it is doubtful how willing all these companies will be to pass from their approach to the e-services approach. At the moment, it seems that standards are useful if you start building an application from scratch. If you already have some applications developed e.g. using an EAI system, it's very difficult to integrate into it e-services using UDDI, as you need to write a lot of integration code, build up some wrapper and so on. In other words, for already deployed applications, the practical solutions used at this moment, seem to be better and simpler than the standards.

Brokering: There may be multiple service providers that offer the same or similar services, i.e., their services can be used to achieve the same or similar objectives. Normally a customer may use the services that offer the most favorable terms in achieving its business objectives. To select the best collection of services, a service integrator may perform dynamic service selection and integration to dynamically construct a new service. This task is heavily dependent on the way the services are modeled and described. There are preliminary research contributions in the area of service quality and automatic service selection via service brokering in the literature [35, 36, 37, 38].

Reliability: Some e-service providers will be more reliable than others. How can this reliability be measured and communicated? What happens when an e-service provider

goes off-line temporarily in situations such as temporary outages caused by nightly maintenance or backups? Do you locate and use an alternative service hosted by a different provider, or do you wait around for the original one to return? How do you know which other providers to trust? What happens if you're in the middle of a payment transaction for the acquired e-service? On the other hand, if you are the company that makes direct Web service connections possible, how do you achieve disaster recovery and migration of all your business partners to a backup system? How do you describe reliability and how do you ensure that a service satisfies your needs? How QoS in general (parameters such as the throughput, response time, cost) is expressed in a set of parameters? Preliminary research contribution includes the CMI system that implements CMM model [38]. CMM provides that each service has a set of QoS attributes that provide information on the expected QoS of the service execution. Furthermore, each activity variable in a process type that is bound to a service interface can have a description of the desired QoS. This description can include conditional statements to allow adaptation of the favored QoS to the current situation of the running process. Thus, if there are multiple service providers, the provider that is selected is the one who offers a service that best matches the desired QoS goal.

Security: Some e-services will be publicly available and unsecured, but most business-related services will need to use encrypted communications with authentication. It is likely that HTTP over SSL will provide basic security, but individual services will need a higher level of granularity. Do services need to be able to provide security at the method level? How does an e-service authenticate users? If you sign up with a provider that provides services around the world, how do these services learn about your security privileges? Proposed standards such as XML Key Management Specification (XKMS) [39], XML Signature, XML encryption and OASIS Security services seems to be able to be incorporated into e-services systems.

*Transaction*s*:* Traditional transaction systems use a two-phase commit approach – all of the participating resources are gathered and locked until the entire transaction completes, at which point, the resources are finally released. This approach works fine in a closed environment where transactions are short-lived, but doesn't work well in an open environment where transactions can span hours or even days. What kind of transaction should be integrated into E-services? It seems that the proposed standards such as XAML [40] and XLANG [41, 42] provide some solutions in these questions. More specifically, XAML has been designed to support current transaction monitors and includes support for commit, cancel, and retry operations. For example, one can ask for commitment among all web services involved in a business transaction. If one fails or refuses to commit, XAML provides the framework for canceling the operation. XAML can ensure transaction integrity and provide a single, complete business transaction to the consumer. XLANG is based on an optimistic model that has been proposed in database research, where actions have explicit compensatory actions that negate the effect of the action. XLANG is a notation for expressing the compensatory actions for any request that needs to be underdone.

Manageability: What kinds of mechanisms are required for managing a highly distributed system [43]? Since the properties of the system are a function of the

properties of its parts, do the managers of each of the various e-services need to coordinate in a particular way? Is it possible to "outsource" the management of some e-services to other e-services? Initial research contribution in this aspect is provided by the Common Information Model (CIM) [44] that defines the schema and protocol standards for enabling the design of management e-services.
Accountability: How do you define how long a user can access and execute an e-service? How do you charge for e-services? Will the dominant model be subscription-based, or pay-as-you-go? If you sell an e-service, how do you designate the ownership has changed? Can an e-service be totally consumed on use, or can you reuse the service multiple times as part of your purchase agreement? Standards do not provide any answers to these questions and, to our knowledge, there are yet not any research results.

Testing: When a system is comprised of many e-services whose location and qualities are potentially dynamic, testing and debugging takes on a whole new dimension. How do you achieve predictable response times? How e-services that come from different providers, hosted in different environments and on different operating systems can be debugged? These questions are still open.

E-service Monitoring and Analysis: How service executions can be monitored and how service execution data can be analyzed in order to improve the service quality or efficiency. An answer to this is provided by eFlow [29], as it includes components that allow users to monitor and analyze a service while in execution.

E-Service Contracts: A contract specifies, usually in measurable terms, what services the provider will furnish. Some metrics that contracts may specify include: what percentage of the time services will be available, the number of users that can be served simultaneously, the schedule for notification in advance of changes that may affect users and help desk response time for various classes of problems. In ebXML a Trading Partner Agreement (TPA) [21] is created during an engagement between two parties. It contains information that outlines the service and the process requirements agreed upon by all parties. The TpaML (Trading Partner Markup Language) [45] is an additional specification proposed by IBM to manage conversations between trading partners.

5 Conclusion

E-services is the next stage of evolution for e-business. Perhaps what is the most intriguing about the e-service paradigm is that what it matters is the e-service functionality irrespectively of the technology that has been used to build them. An e-service is accessible over the web, it exposes an XML interface, it is registered and can be located through an e-service registry and it communicates with other services using XML messages over standard Web protocols. Therefore, all web services environments can interoperate – at least in theory.

There are a number of standards, frameworks and tools that support e-services but it seems that they still are in early stages of development. In this paper we have given an overview of evolving standards, such as WSDL, SOAP and UDDI and we

examined the various technical challenges and the corresponding research contributions. From this analysis, it became obvious that the examined existing standards approach e-service development bottom-up and provide solutions only to problems at a very low abstraction level. They don't tackle problems existing at a high abstraction level, e.g. how a business user can describe his/her needs? How can s/he select the optimum service from a number of different providers? How can s/he be sure that the selected e-service indeed satisfies his/her needs? We consider that answers to these questions are of first priority as, there is not any use to know the format of messages of an e-service if you don't know if this service satisfies your needs, need. We examined some original research contributions that address the e-service development using a top-down approach in this area e.g. the CMM project [38]. We think that what is needed is a merge between these solutions and the existing standards.

It seems that E-services may be an evolutionary step in designing distributed applications, however, they are some issues that require careful consideration. There are a lot of hurdles and limitations that must be overcome in order for mass adoption to occur. It is important that all initiatives cooperate in the development of universally accepted e-services standards, because one of the key attributes of Internet standards is that they focus on protocols and not on implementations. Otherwise, competing standards from industry heavyweights could prevent widespread adoption of e-services.

References

1. http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomarch.htm
2. http://www.sun.com/jini
3. http://java.sun.com/products/ejb/
4. E-services: Taking e-business to the next level. IBM White Paper, Available at http://www-3.ibm.com/services/uddi/papers/e-businessj.pdf
5. Fingar, P.: Component-Based Frameworks for E-commerce. In Communications of the ACM, Vol. 43 (10), (2000), 61-70
6. Web Services architecture overview: The next stage of evolution for e-business. Available at: http://www-106.ibm.com/developerworks/library/w-ovr/?dwzone=ws
7. http://208.185.204.196/esv/home.htm
8. http://uddi.microsoft.com/
9. https://service.ariba.com/UDDIProcessor.aw
10. http://www-3.ibm.com/services/uddi/
11. www.xmethods.com
12. http://msdn.microsoft.com/xml/general/wsdl.asp
13. http://www.soap-wrc.com/webservices/default.asp
14. http://www.uddi.org
15. http://www.oasis-open.org/cover/uddi.html
16. Kuno, K.A.: Surveying the E-services Technical Landscape. In Proceedings of the 2nd Int. Workshop on Advance Issues of E-Commerce and Web-Based Inf. Systems (WECWIS 2000), IEEE Computer Society, (2000), 99-10. On line proceedings at: http://www.computer.org/proceedings/wecwis/0610/0610toc.htm
17. Pelzt, C.: Interacting with services on the Web: a review of emerging technologies and standards for e-services. Available at: http://devresource.hp.com/devresource/ Topics/

18. van den Heuvel, W.J., Yang, J., Papazoglou, M.P.:Service Representation, Discovery and Composition for E-Marketplaces. To be presented at the 6th Int. Conf. on Cooperative Information Systems (CoopIS 2001), Trento, Italy, September 5-7, (2001)
19. The Advertisement and Discovery of Services (ADS) protocol for Web Services. Available at : http://www.cn.ibm.com/developerWorks/web/ws-ads/index_eng.shtml
20. ebXML – White Paper – Enabling Electronic Business with ebXML. Available at: http://www.ebxml.org/white_papers/whitepaper.htm
21. Collaboration-Protocol Profile and Agreement Specification v1.0, Available at: http://www.ebxml.org/specs/index.htm
22. Kim, W., Graupner, S., Sahai, A., Lenkov, D.: E-speak – a XML Document Interchange Engine for Web-based e-Services. To be presented at the *EC-WEB 2001 Conference*, Munich, Germany, 3-7 Sept. (2001)
23. Hewlett-Packard, eFlow Model and Architecture, version 1.0. (1999)
24. Registry Information Model v1.0, http://www.ebxml.org/specs/index.htm
25. Message Service Specification v1.0, http://www.ebxml.org/specs/index.htm
26. Yang, J., van den Heuvel, W.J., Papazoglou, M.P.: Service Deployment for Virtual Enterprises. In Proceedings of ITVE 2001 Workshop on Information Technology for Virtual Enterprises, Queensland, Australia, Jan. 29-30, (2001)
27. http://www.w3.org/2000/xp/Activity
28. Casati, F. et al: Adaptive and Dynamic Service Composition in *eFlow*. Available at: http://www.hpl.hp.com/techreports/2000/HPL-2000-39.html
29. Casati, F., et al.: eFlow: a Platform for Developing and Managing Composite e-Services. In Proceedings of the Academia/Industry Working Conference on Research Challenges (AIWoRC 2000), Buffalo NY, April 27-29, IEEE Comp. Society, (2000)
30. Schuster, H., Georgakopoulos, D., Cichocki, A., Baker, D.: Modeling and Composing Service-Based and Reference Process-Based Multi-enterprise Processes. In Proceedings of CaiSE 2000. Lecture Notes in Computer Science, Vol. 1789, Springer-Verlag, Berlin Heidelberg New York (2000) 247-263
31. http://www.tibco.com/products/index.html
32. http://www.seebeyond.com/
33. http://www.bea.com/products/elink/index.shtml
34. http://www.webmethods.com/
35. Geppert, A., Kradolfer, M., Tombros, D.: Market-Based Workflow Management. In International Journal of Cooperative Information Systems, Vol. 7 (4), (1998)
36. Bichler, M., Segev , A., Bean, C.: An Electronic Broker for Business-to-Business Electronic e-Commerce in the Internet. In International Journal of Cooperative Information Systems, Vol. 7 (4), (1998)
37. Nodine, M.H., Bohrer, W., Ngu, A.H.H.: Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In Proceedings of ICDE (1999) 358-365
38. Georgakopoulos, D., Schuster, H., Chicocki, A., Baker, D.: Managing process and service fusion in virtual enterprises. In Information Systems, Vol. 24 (6), (1999) 429-456
39. http://xml.coverpages.org/xkms.html
40. http://www.xaml.org/
41. http://entmag.com/displayarticle.asp?ID=11220072307AM
42. http://www.xmlmag.com/upload/free/features/xml/2000/04fal00/sg0004/sg0004.asp
43. Machiraju, V., Dekhil, M., Griss, M., Wurster, K.: E-Services Management Requirements. In Proceedings of Technologies for E-services Workshop, Cairo, Egypt. September (2000)
44. CIM Standards, Available at: http://www.dmtf.org/spec/cims.html
45. TpaML Specification, Available at: http://www.oasis-open.org/cover/tpa.html