The Internet Enterprise

Sumi Helal, Stanley Su, Jie Meng, Raja Krithivasan, and Arun Jagatheesan

Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611, USA
Helal@cise.ufl.edu

Abstract

In this paper we present our vision of the Internet Enterprise: a highly interoperable, virtual enterprise infrastructure for individuals, small businesses, and large corporations. In the Internet Enterprise (IE), any service provider can enable its business to be programmatically accessible on the Internet (becomes an e-service). Enabled by a scalable brokering service and an inter-organizational workflow facility, e-services offered by autonomous service providers could be composed into an "Internet Workflow". This paper describes this vision through an architecture that treats e-services as workflow participants in Internet-wide workflow automation applications. We present the architecture and implementation of three core components that enable the Internet Enterprise. These are: (1) BizBuilder, an e-service framework, (2) Sangam, a scalable, hierarchical brokering community based on UDDI, and (3) a dynamic workflow engine that uses e-services as entities, to create and enact workflow models. Collectively, these three components empower the Internet as a "public enterprise", where virtually any person or organization can design workflow models, and hence create new businesses, using available, competing e-services.

1. Introduction

The phenomenal growth of business-to-business (B2B) e-commerce has fueled the development of software systems that would integrate themselves seamlessly into the existing B2B space and provide better value-added services. This rise in the growth of B2B e-commerce is a natural outcome of the success of the earlier model, the business-to-consumer (B2C) e-commerce. It was soon realized that the Internet infrastructure could be used by businesses effectively as done by consumers. In other words, businesses would use Internet as their communication medium to set-up business deals and utilize the services they need from other businesses, just in time when needed. A supply chain consisting of a manufacturer, distributor and seller can be taken as an example where a business requires the service of another.

The most fundamental requirement for success in the Business-to-Business e-commerce domain is the ability to seamlessly integrate and/or inter-operate diverse businesses. In other words, services offered by one business should be easily accessible to any consumer or other business using the Internet infrastructure and standard protocols. The Internet Enterprise project addresses this very requirement by providing frameworks and core components that enable: (1) the creation of new e-services or the conversion of existing businesses into e-services, (2) the advertisement and discovery of relevant e-services within a scalable brokering community, and (3) the composition of multiple e-services into workflow models and new, composite e-services.

In this paper we describe the Internet Enterprise architecture and show how it can be used to create "Internet Workflows". The architecture consists of BizBuilder, our e-service framework, Sangam, our brokering community framework, and a dynamic workflow server used to design and enact workflow models based on e-services. BizBuilder addresses the issue of workflow in particular and facilitates managing both synchronous and long-running e-services that participate in a workflow. Additionally, BizBuilder provides suitable support to register e-services to the Sangam broker, which is an Universal Description, Discovery and Integration – UDDI enabled brokering community. A workflow server uses the Sangam brokering services twice: once at design time for discovery of relevant e-services, and later during execution time to bind e-service requests to actual service providers.

The paper is organized as follows. Section 2 summarizes related work and contrasts it to our approach. Section 3 presents the highlights of the Internet Enterprise architecture. Section 4 gives the details of BizBuilder. Section 5 gives the details of the Sangam broker. Section 6 gives the details of the workflow server and shows a simple example of workflow modeling and enactment. Finally, our conclusions and future work are summarized in section 7.

2. Related Works

The term *workflow* originated in the mid-1980s and became popular in the early ’90s. Since then, many vendors have offered general-purpose workflow management systems [MQWF][VIRI]. In addition to the general-purpose workflow management systems, some other fields co-opt workflow capabilities [Sheth99]. For example, most leading

Enterprise Resource Planning (ERP) systems have a workflow component. Another field that adapts workflow functionality is Enterprise Application Integration (EAI) [Oba01]

Recently, with the emergence of e-business and virtual enterprise, the workflow technology has become hot again. Currently, many research efforts are trying to solve the problems and challenges that this new area has brought to the workflow technology.

WISE (Workflow based Internet SErvices) is a project conducted at the Swiss Federal Institute of Technology [Lazc00] [Alon99]. It aims to design, build, and test a commercially viable infrastructure for developing distributed applications over the Internet. The infrastructure provides an Internet-based workflow engine acting as the underlying distributed operating system for controlling the execution of distributed applications, and a process modeling tool for defining and monitoring the process. CrossFlow is a European research project for supporting cross-organizational workflow management in virtual enterprises [Gref98]. Its goal is to develop and implement a mechanism for connecting WfMS and other WfMS-like systems of different organizations in cross-organizational workflows and electronic commerce settings. CrossFlow defines a service-oriented model for cross-organizational workflows. In their service-oriented model, a service specifies which part of workflow it fulfills. The service provider of each service can be either an internal resource (internal service) or an external organization (external service). For an external service, service selection at run-time will be based on the QoS parameters given in service specifications.

Our Internet workflow system integrates autonomous e-services provided by the organizations across Internet. Unlike the service definition in CrossFlow, the e-services in our workflow system are defined and provided independent of business process models.

3. The Internet Enterprise Architecture

The overall architecture of Internet Enterprise is shown in Figure 1.

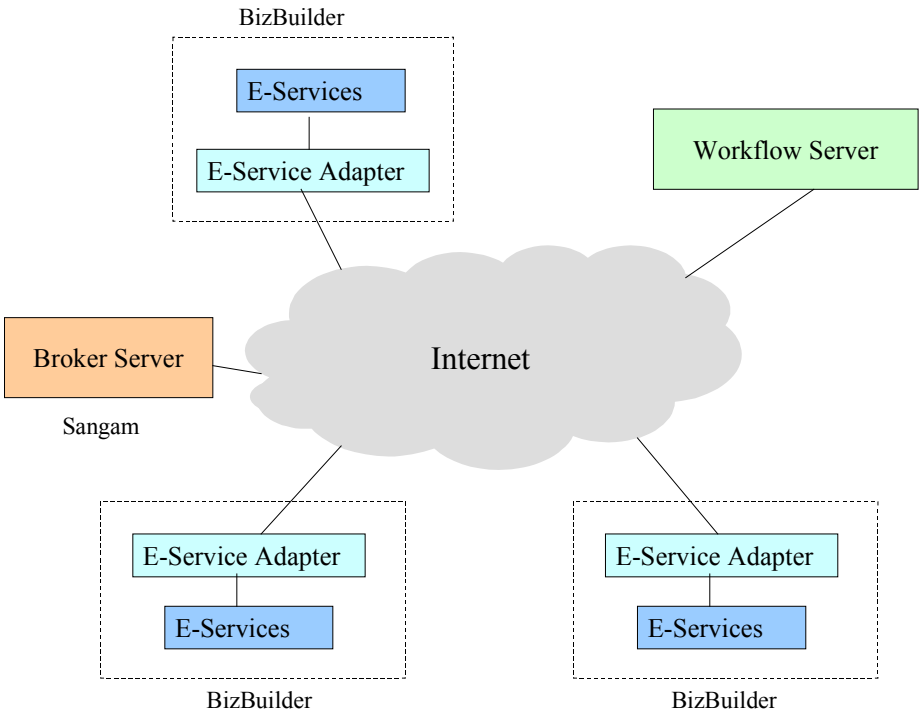

Figure 1. The Internet Enterprise Architecture

Business organizations (small business or large corporations) across the Internet can perform and contribute different manual or automated tasks, which are useful for the operation of a joint business. These tasks can be treated uniformly as e-services. An e-service adapter needs to be installed at each organization's site as a wrapper of its services so that they can be accessible on the Internet and participate in an Internet-based workflow.

A Broker Server is used in the system to manage the e-service specifications that have been registered by participating business organizations and to match e-service requests against these specifications for selecting the suitable e-service providers [Robi98].

The Workflow Server is composed of two sub-components: namely, the Process Definition Tool and the Workflow Engine. The Process Definition Tool is responsible for modeling business processes, which integrate the e-services across the Internet. The

Workflow Engine schedules the enactment of business processes according to the process model specification. E-service requests are specified in a process model according to some standardized *e-service templates* that are managed by the Broker Server, and are bound to the proper service providers at run-time by using the services of the Broker Server to identify the suitable providers. E-service requests are then sent to corresponding e-service adapters to invoke e-services.

4. BizBuilder

4.1 E-Services

E-services in our view are, services that are offered on the Internet, which are independent, self-describable entities of value, and which can be accessed *programmatically* using standard Internet Protocol and representation formats like HTTP and XML. E-services transform traditional services (which are mostly system-dependent) into inter-operable entities that hide system-dependent interactions. In other words, E-services can be accessed in a uniform way, irrespective of the type or the system used to implement these services. This inter-operability [Dan98] is the key feature that enables E-services to be used in Internet-based applications. E-services are significant entities of value in the business-to-business e-commerce market space, where an array of similar services exists simultaneously and any one of them can be accessed anytime based on the quality of service, without prior planning. In such a scenario an array of E-services can be utilized seamlessly, since all of these E-services by definition, are inter-operable and can be used just in time when needed. We believe that an Internet-based workflow can utilize E-services as the individual elements, which can suitably be grouped together to execute a workflow of tasks.

In the present scenario, most services are accessed using a homogeneous infrastructure similar to a service provider, or these services have suitable web-interfaces (using HTML web-pages and applets) so that they can be accessed on the Internet. These services cannot be termed as E-services, since they lack the fundamental properties of E-services, namely, they cannot be accessed in a uniform way by a variety of systems, nor they can be described suitably. We have designed BizBuilder – an E-services framework

that provides the necessary tools using which E-services can be created and utilized. In specific terms, the framework provides the facility to

- Take an existing object (or service) implementation and provide an E-service wrapper
- Take an existing object implementation and provide a XML interface description of an object (to represent an E-service)
- Provide the facility for advertising a service to a UDDI enabled broker
- Provide necessary tools and APIs to invoke an E-service on the Internet
- Provide support for the E-services to participate in a workflow

4.2 Components of the BizBuilder Architecture

The fundamental objective of the framework is to facilitate the creation of these E-services, suitably represent them, so that they can be utilized in a system-independent way using HTTP and XML. The framework allows existing services (implemented using java objects) to be suitably wrapped, so that they can be exposed as E-services in an inter-operable environment. The architectural components of BizBuilder are depicted below.

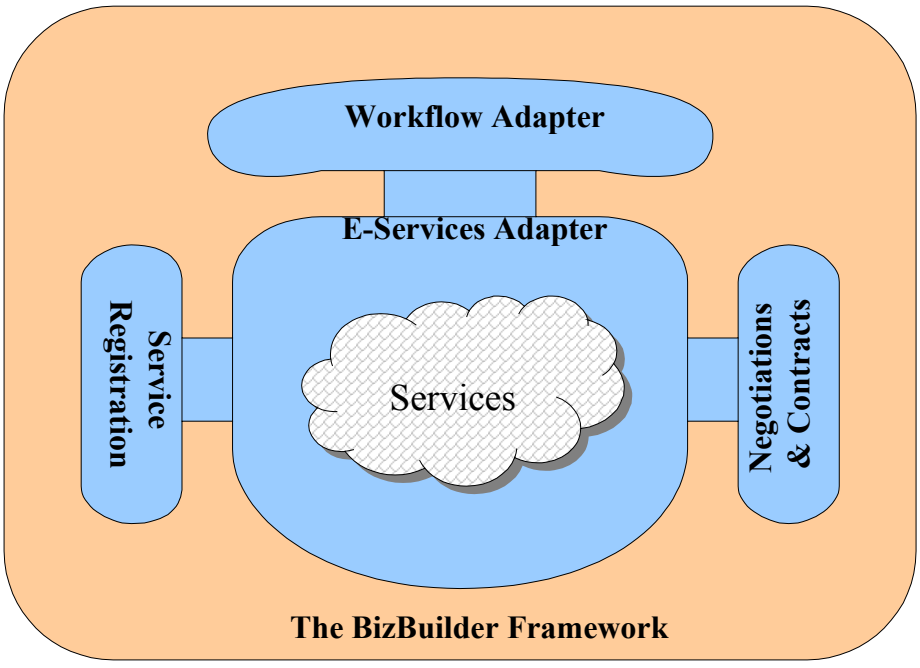

Figure 2: Components of the BizBuilder Framework

4.3 The E-Service Adapter

Data Exchange in E-services

One of the foremost goals of E-services is to enable them to be accessed in a system or programming language independent way. Therefore the data exchange that is done during an E-service invocation should follow a scheme that is easily adoptable and language-neutral. On the other hand the E-services by themselves are remote services, they are provided by the service provider and are utilized when needed. So a message format for data exchange is needed that will facilitate conveying the required data to these remote services. The above reasons made us choose SOAP [Boz00]- Simple Object Access Protocol for data exchange and for E-service invocation. SOAP specifically addresses the issue of RPC over XML and is extensible since its based on XML. So the data exchange done during an E-service invocation (during a service request and reply) follows the SOAP message format.

E-Service Representation

E-services have to be suitably represented, so that they can be easily searched for, and compared during service discovery. The representation also needs to be rich, in order to describe the various aspects of service, like nature or type of service, category of the service or the business domain it belongs to, the data required to invoke the service, type of results returned, service access points etc. The WSDL [Chr01] – Web Services Description Language specifically addresses the issue of service representation and our E-services are represented using WSDL. The BizBuilder framework provides the suitable interfaces to register or advertise these E-services to a UDDI [UDD00] enabled broker. The broker serves as a central repository for service advertisement and is typically used by the service inquirers to find a required service of their interest and relevance. The service representation is detailed later along with the broker components in section 5.3.

The E-service Adapter

The E-services Adapter is the core component that wraps the underlying services (for example services implemented using java objects) as E-services. This module takes

an existing service implementation, that could have been implemented in different ways, using java objects, or using a standard workflow methodology or on a legacy system, and creates the required server side framework, so that these E-services can be invoked using a XML messaging structure on the HTTP protocol. The E-services adapter thus hides the heterogeneity of service implementation and presents a uniform view of these services as E-services.

The E-service Adapter is the component that is responsible for performing the actual service invocation on the underlying object. The adapter deals with functions like converting the XML message request into suitable native method invocation call and performing them. In specific terms, the E-service Adapter parses the SOAP request and determines the service to be invoked. Using mapping information that is made available, the E-service Adapter instantiates, suitable objects on which the method invocation should be performed. It then builds the parameter list as required by the service and invokes the service. The E-service Adapter deals with issues like object creation and management, parameter type identification and creating objects of suitable types from the SOAP coded request. So the E-services adapter is the core component that performs the actual method invocation on the underlying object, hiding the internal details from the abstract E-service representation.

4.4 The Workflow Adapter

Long Running E-services

Services that are offered on the Internet differ significantly from one another by their nature. For example services like document proofreading, requiring manual intervention, can be termed as “long running services”. Long-running services are basically asynchronous services, which may take an arbitrary time for completion of execution. The E-services adapter determines the long-running services and handles them suitably. These long-running services can participate in a workflow, in which case required interfaces have to be provided, so that a workflow (utilizing this service) or the user of this service can query the status of execution of these services. The workflow adapter module provides these necessary workflow query capabilities.

Workflow Adapter

The ability to make E-services participate in a workflow and allow other service inquirers or users to utilize an E-service is one of the main objectives of BizBuilder. In order to achieve this, the framework has to provide a way for the service inquirers to query about the status of execution of an E-service, so that they can manage and control their workflow. The workflow adapter component is the workflow support interface intended for the service inquirers.

Support for Workflow Probes

An E-service should be able to participate in a workflow in which case it should support workflow specific queries like "how much of the service is done", " What is the expected completion time", "what is the additional cost for early completion by mm/dd/yyyy" etc. These types of queries can be raised by a workflow engine, which controls the execution of a workflow instance (enactment of a business process) that uses the service under discussion to perform a part of an activity within a workflow. Such queries and their result help the workflow engine to dynamically modify or manage the execution of workflow. This sort of scenario can be applied to a fault-tolerant and a critical workflow process in which a failure in one service or activity of a complex workflow should not affect the whole workflow as such, instead the workflow should be able to dynamically find and utilize an alternate service.

The Workflow Adapter provides the necessary functionalities to handle these workflow queries, which are similar to any other normal service implemented by the service object. But in this case, workflow specific queries have to be uniquely identified, and suitable responses in this case have to be created. It is also important to mention here that all workflow queries have to be ultimately answered by the service provider objects, since the objects are the ones that actually implement the service and know the status of the execution. BizBuilder provides a workflow interface, which a user can extend and implement, to support workflow probes. Additionally, BizBuilder provides the messaging infrastructure to query the service objects (that should have already implemented the workflow interfaces) using the workflow probes.

The workflow interface supports probes like,

- *getStatus* – used to get the current status of execution of the task or service, typical return values include status like "expected completion in 2 hours"
- *isComplete* – used to query if the service has completed executing or return the percentage of task completed, typical return value is "80% completed"
- *query* – a general-purpose query message intended for the task (example, can the task be finished in 1 hour). These types of messages, can be used to interact with the workflow enabled service, during execution to perform activities like re-negotiation
- *tell* – intended as a one-way notification message to the activity or the task
- *commit* – commit the current execution of the task or the service
- *abort* – abort the current execution of the task or the service
- *getResult* – get the result of the current execution of the service or the activity
- *isSynchronous* – used to determine whether a service is synchronous or long-running

4.5 Negotiation and Contracts Adapter

In order to provide a complete E-services framework, BizBuilder also includes support for Negotiation and Contracts. This module provides a simple negotiation framework, where the service inquirers can negotiate a particular E-service for usage. All negotiations will be carried out on a per-service, per-client basis. The Negotiation protocol is a synchronous protocol, in which the negotiating parties use a pre-defined data format to communicate messages. Once an agreement is reached upon the criterion in question (like cost, quantity, response time etc), a contract is established and the service inquirers are bound to the contract. Support to modify existing contracts will be provided by these modules and the contracts once created are stored in persistence storage to be used later during actual service invocation.

5. Broker Server

In the previous section we saw the uses of e-services and how they can be built. In order to use the emerging e-services to create Internet based workflow, we need to have well defined description, discovery and integration of e-services. In this section, we

find a detail description of how services are discovered in a dynamic environment using the brokering service [Hela01].

5.1. Relationship between the Broker Server and other Components

Figure 3 shows the relationship between the Broker Server and other components. The Service Provider is the owner of a service, which is deployed as an e-service. The Broker Server maintains a local node of UDDI [Bou00] based registry repository containing service descriptions from the service providers. The Workflow Server is a service requestor looking for appropriate service providers.

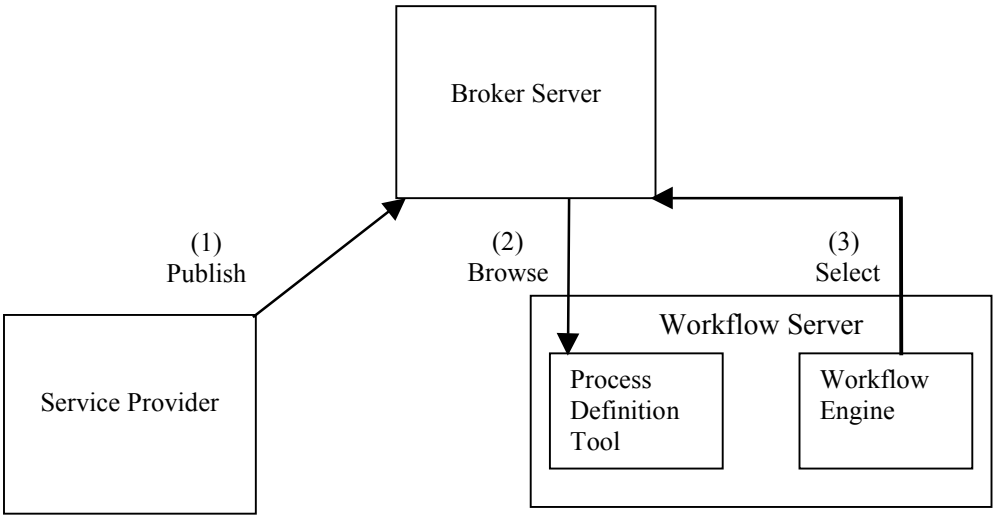

Figure 3. Relationship between the Broker Server and other components

The operations performed in Figure 3 are:

- *Publish/Unpublish*. Service providers advertise (publish) their e-services to one or more broker servers. They may also unpublish the advertisements of service from the registry.
- *Browse*. The designer of the workflow browses *service templates* available in the Broker Server to find the service templates that can be used in the design of the *process models* (explained in the next section).
- *Select.* At run time the Workflow engine makes queries to the Broker server to get desired service provider(s) for specified e-service request(s). The e-service requests are defined based on e-service template(s). An e-service request may

also have constraints that require the service provider to have capabilities that can satisfy the constraints. The Broker Server uses its knowledge of the service providers and to find the possible binding information for the Workflow.

5.3. E-service Description using WSDL

Description of e-services is very important as it helps the service to be discovered. An e-service description has to provide the required information on how to bind and invoke the service. Also it must have a high level information about the service and its category.

In our implementation, we use WSDL (Web Services Description Language) [Chr01] to describe the e-service interfaces. WSDL is essentially an XML IDL (Interface Definition Language) that can describe the functions and interface of a service. It is an XML format for describing network services as a set of endpoints operating on messages contains either document-oriented or procedure-oriented information. WSDL allows the operations and messages to be described abstractly and then bound to concrete protocol and end point. WSDL is extensible to allow description of endpoints.

The service description is divided into two parts: *E-Service Template Description* and the *E-Service Binding Description*. The *E-Service Template Description* gives description on the methods and arguments that of a service. This information will be needed by workflow developer to programmatically use the e-service. The *E-Service Binding Description* contains the location of the service implementation and details on protocol and port to be used to access the server, which hosts the service.

WSDL allows specification of both the e-service template and the binding information. The template description and the binding description are separated as two WSDL documents – *WSDL template document* and *WSDL binding document*. This is advantageous, as the service providers need to only provide the WSDL binding to access their service (if a WSDL template document exists already in the broker for the service). Also, the workflow engine has to know only about the templates and use them in their design time. At run time, the workflow engine can get the binding information required for any service template from the broker.

A WSDL document uses the following elements in the definition of network services:

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

It is important to observe that WSDL is not a definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification. WSDL allows using other type definition languages via extensibility.

5.4 Service Discovery for Workflow

In this sub-section we describe all the processes involved in a service discovery step-by-step from publication till binding used for workflow. We will start with the publication of e-service template.

Publish E-service Template Description

E-service template description is an abstract definition of the e-service. In our implementation we use WSDL template document for e-service template description. WSDL template document contains the data types and messages used in an e-service. The service providers require e-service templates to define their services based on a standard. If a service provider is not able to find a service template that can be used to define a new service, he can create a new service template using the following steps.

1. *Define E-Service Template Description.* An E-Service template describes the interface of the service.

2. *Determine Category.* The category of the e-service has to be determined. The broker server in our case uses NAICS (North American Industry Classification System) internally for the classification of services. The broker service can internally map the categories and the users need not lookup for service category codes.
3. *Publish to Broker.* The e-service template description is published in the broker. The publisher of the e-service template has to just supply the URI information of the WSDL Template document. The broker reads the WSDL document, validates it and updates it to the registry it maintains.

Publish E-Service Binding Description

Once an e-service is ready to be published, service provider can use SOAP messages to publish its e-service binding description to the broker. The steps involved here assume that the service provider information and the e-service template description are already published in the registry.

1. *Determine Category.* The publisher of the e-service has to determine to which category the e-service belongs. The "getCategories" message is used to contact with the broker.
2. *Determine E-Service Template Description.* The WSDL template document based on which the e-service implementation has been made has to be found from the broker. The message sent to the broker is "getInterface" SOAP request.
3. *Generate E-Service Binding Description.* A WSDL binding document describing the binding protocols and ports with the URI's for the e-service implementation has to be prepared. Also, a service provider may embed his constraints on the input and output (attributes) of the service in the binding document. Inter-attribute constraints can also be specified. Constraints are necessary for the brokering service to rank the matches [SU87] between he service provider and the requesting service queries. The Broker Server uses a Constraint Satisfaction Processor for this [HAM98].
4. *Publish E-Service.* The WSDL Binding Document containing the e-service binding description is published using the SOAP message "putService".

Build E- Service Requests

The process model designer needs to build e-service requests during the process modeling. He/she discovers an e-service and uses it following steps below.

1. *Determine E-Service Template Description.* The process model designer has to first find e-service templates before building the e-service requests. The e-service template contains the attributes and operations that the service implementations of different service providers must support and the arguments that are used in the service. The "getInterface" message is used to contact the broker.
2. *Build E-service Requests.* Based on the information received from the e-service template, the input data needed to use an e-service are specified in e-service requests. The process model designer can also define constraints on the attributes of e-services request.

Select Service Provider (s)

At runtime, the Workflow Engine needs to contact the Broker Server to get service providers for e-service requests in a process model.

1. Use the service at Runtime. At Runtime, the Workflow Engine contacts the broker with e-service requests to select service provider (s) whose e-service implementations satisfy these e-service requests. Also the Workflow engine may provide some constraints that will have to checked by the broker for matching with the service provider's binding description. This query gives the URI of the service provider that matches with the constraints.

Now that we have seen how the usage of the broker service, in the next section we see how take a look at the operation of the workflow engine in creating dynamic workflow.

6. Workflow Server

The Workflow Server consists of two parts: the Process Definition Tool, which is used to the process modeling at build-time, and the Workflow Engine, which is responsible for the scheduling the execution of business processes at run-time.

6.1 Process Modeling

The e-service requests are building blocks for the process models in Internet-based Enterprise. Basically, a process model is a network of activities that contains e-service requests. The Process Definition Tool is a user-friendly graphical editor for creating process models. It can be used to specify the diagram of a business process model, data objects manipulated by the business process, and the e-service requests details. We now discuss how to specify e-service requests in a process model.

An organization can provide multiple e-services and an e-service can be provided by multiple organizations. In the Internet environment, providers of e-services may change frequently; new providers are added and old providers become unavailable. In modeling business processes, it is therefore important to separate e-service requests specified in a process model from their providers. That is, a process model should not statically bind its e-service requests to specific providers at the time a process model is defined. The binding should occur at run-time when the available providers are known to the workflow management system.

During process modeling, the *e-service requests* specified in a process model are defined in terms of the attributes given in their corresponding e-service templates. An e-service template consists of the following three general types of attributes:

- Input attributes, which specify the data needed as input to invoke an e-service.
- Output attributes, which specify the returned data of an e-service.
- Service attributes, which specify other properties of an e-service, such as the length of time the e-service takes, the cost for using the e-service and the quality of the e-service, etc.

An example of the e-service template for an e-service provided by business type *Distributor* is shown in Table 1.

Table 1. E-Service Template of e-service *Process Order* of *Distributor*

E-service	Description			
			Name	Type
Process Order	E-Service Attributes	Input Attributes	Product_Name Model_Name Quantity User_Info	String String Int UserInfo
		Output Attributes	Order_Status	Status
		Service Attributes	Duration Cost	Time Float

In a process model, an e-service request defined according to its corresponding e-service template consists of three parts: the input attributes mapping information, which map the output attributes mapping information, and the constraints on the service attributes can also be specified in an e-service request. The specification of the e-service request is shown as follows.

```
SERVICE  <service name>
            INPUT     <input attributes mapping>
            [OUTPUT   <output attributes mapping>]
            CONSTRAINT <e-service request constraint>
 END_SERVICE
```

We shall call the constraints in an e-service request *e-service request constraints*. An example of e-service request constraint is shown below.

```
ATTRIBUTE_CONSTRAINT:
      duration          int        [0 .. 10]          priority[1]
      cost              float      [0 .. 1000]        priority[2]
INTER_ATTRIBUTE_CONSTRAINT:
            lac1   duration >4   implies   cost < 800
```

The above constraint specification states that the requester of this e-service request expects that the e-service should not take more than 10 time units, the cost of the

e-service should not be more than $1,000, and if it takes more than 4 time units, then the cost must be less than $800.

6.2 Workflow Enactment

The enactment of a business process is performed by the Workflow Engine, which forms the core of the run-time environment. The run-time interactions between the Workflow Engine and other components of Internet Enterprise are shown in Figure 4.

The Workflow Engine schedules the execution of a workflow instance based on the process model. When an e-service request is scheduled for execution, the Workflow Engine would contact the Broker Server to bind the e-service requests to the services of suitable service providers in a dynamic service binding process.

The dynamic service binding process would make use of the constraint-based service provider selection function that is provided by the Broker Server. To achieve constraint-based service provider selection, the Broker Server would match an e-service request with e-service descriptions given by service providers to identify the proper service providers for the request. The data provided for the input attributes of an e-service and the constraints specified in the request would have to match with (i.e., not conflict with) the attribute constraints and inter-attribute constraints specified in the e-service binding descriptions by the service providers.

To invoke an e-service, the Workflow Engine would generate the SOAP XML message containing e-service request information and send a SOAP XML message to the E-service Adapter at the site of the selected service provider. A SOAP XML message containing the response information is returned to the Workflow Engine after the e-service is performed.

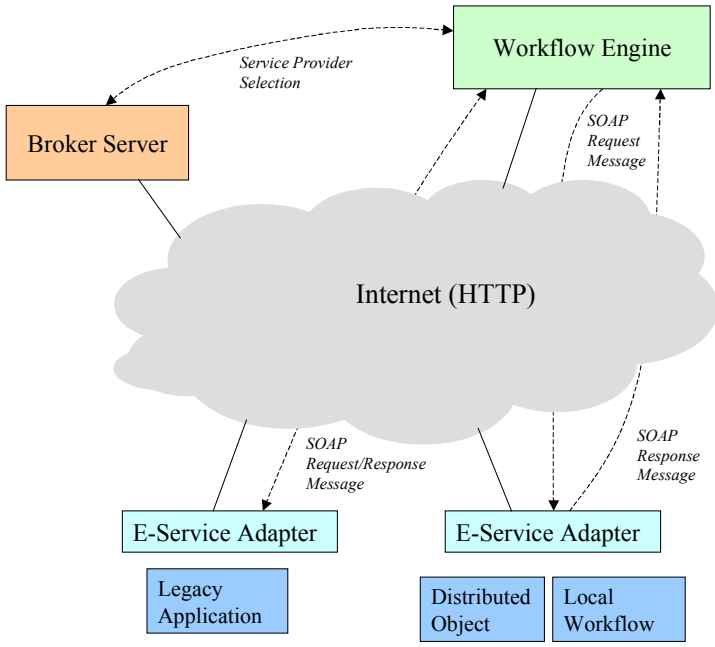

Figure 4. Run-time Interaction between the Workflow Engine and Other Components

The main components of the Workflow Engine are a Workflow Scheduler, an Activity Handler, and a Service Handler. The Workflow Scheduler schedules the transitions between activities, and thus controls the execution of a workflow instance. When the Workflow Scheduler determines that an activity is ready to be executed, it delegates the execution of this activity to the Activity Handler. After an activity is completed, the Workflow Scheduler would decide the next activity to be processed. The Activity Handler is responsible for executing the activity and contacting the Broker Server to dynamically bind e-service requests in the activity to a suitable service provider. The URL address of the E-service Adapter for this service provider is then obtained. The Service Handler is responsible for invoking the remote e-services. For each e-service request, it sends the XML message containing the request data to the remote E-Service Adapter, and receives the XML message containing the response data. The architecture of the Workflow Engine is shown in Figure 5.

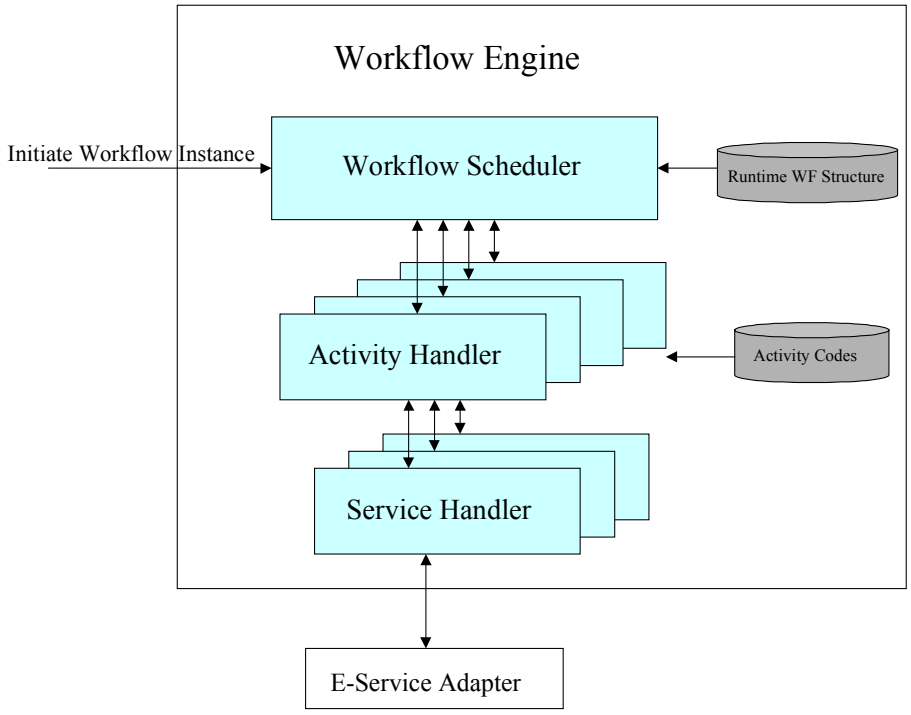

Figure 5. Workflow Engine Architecture

7. Conclusion

In this paper, we have presented our vision of the Internet Enterprise that is enabled by inter-organizational workflows. Workflow is a powerful methodology that can be utilized effectively on the existing Internet infrastructure. E-services are universally accessible individual entities of value. We have used E-services as workflow participants in order to create an Internet-wide workflow system. To complete the framework we also created a hierarchical brokering community based on UDDI.

We have implemented a dynamic workflow engine that uses E-services as entities, to create and enact a workflow. BizBuilder provides the necessary framework to create E-services from existing Java implementation. Our brokering community, currently supports dynamic discovery and categorization of E-services based on UDDI. We are working on extending the support to constraint-based brokering to perform better matchmaking. The workflow engine uses the brokering protocols for discovery of relevant E-services that will be utilized during workflow execution.

8. References

[Alon99] G Alonso, U Fiedler,C Hagen, A Lazcano, H Schuldt, N Weiler, "WISE – Business to Business E-Commerce," Proceedings of 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, Sydney, Australia, March 1999.

[Dan98] A Dan, D Dias, T Nguyen, M Sachs, H Shaikh, R King, S Duri, "The Coyote Project: Framework for Multi-party E-Commerce" Proceeding of the 7th Delos Workshop on Electronic commerce Greece 1998

[Gref98] P Grefen, Y Hoffner, "CrossFlow - Cross-Organizational Workflow Support for Virtual Organizations," Proceedings of 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, Sydney, Australia, March, 1999.

[Ham98] J. Hammer, C B Huang, Y H Huang, C Pluempitiwiriyawej, M Lee, H Li, L Wang, Y Liu, S Y W Su, " The IDEAL Approach to Internet-Based Negotiation for E-Business" Proceedings of the 16th International Conferenece on Data Engineering.

[Lazc00] Lazcano, A., Alonso, G., Schuldt, H., Schuler, C., "The WISE Approach to Electronic Commerce," International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy, Vol. 15, No. 5, 2000.

[Hela01] A Helal, M Wang, A Jagatheesan and R Krithivasan, "Brokering Based Self Organizing E-Service Communities". Proceedings of the Fifth International Sympium on Autonomous Decentralized Systems (ISADS) With an Emphasis on Electronic Commerce, March 26-28, 2001

[Lazc00] A Lazcano, G Alonso, H Schuldt, C Schuler, "The WISE Approach to Electronic Commerce", International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy, Vol. 15, No. 5, September 2000.

[MQWF] IBM, MQSeries Workflow, http://www.software.ibm.com/ /ts/mqseries/workflow/.

[Oba01] M Oba, N Komoda, "Multiple Type Workflow Model for Enterprise Application Integration", Proceedings of the 34thHawaii International Conference on System Sciences, Hawaii, USA, 2001.

[Robi98] Robinson, N William, "Electronic Broker Impacts on the Value of Postponement," Proceedings of the 33rd Hawaii International Conference on

System Sciences.

[Sheth99] A Sheth, W Aalst, I Arpinar, "Process Driving the Networked Economy," IEEE Concurrency, Vol. 7, No. 3; JULY-SEPTEMBER 1999, pp. 18-31.

[Su87] S Y W Su, J Dujmovic, D S Batory, S B Navathe, R Elnicki, "A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems," ACM Transactions on Database Systems, Vol. 12, No.3, September 1987, Pages 472-520.

[Bou00] T Boubez, M Hondo, C Kurt. J Rodriguez , D Rogers "UDDI Programmer's API 1.0"

[Box00] D Box, D Ehnebuske, G Kakivaya, A Layman, N Mendelsohn, H F Nielsen, S Thatte and D Winer, "*Simple Object Access Protocol (SOAP)1.1*" W3C Note May 2000.

[Chr01] E Christensen, F Curbera, G Meredith and S Weerawarana, "Web Services Description Language (WSDL) 1.1" W3C Note March, 2001

[UDD00] Universal Description, Discovery and Integration, http://www.uddi.org.

[VIRI] VITRIA Corporation, Vitria Business Ware, http://www.vitria.com/products/businessware.html.