Managing Next Generation E-Services

Akhil Sahai, Vijay Machiraju, Klaus Wurster
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-2000-120
September 21st, 2000*

E-mail: {asahai | vijaym | kwurster}@hpl.hp.com

Internet, web, management, e-services, XML

The ever-widening reach of Internet has led to the proliferation of e-services. These e-services are sprouting in the form of portals and e-business web sites. Some of these e-services interact amongst themselves and undergo service composition to offer other complex services. Next generation e-services would undertake dynamic service composition because of the multiple benefits this model of operation provides. In addition, these e-services have varied implementations which makes managing these e-services a challenging task. These web based e-services also operate on line and would need continuous monitoring and control. This paper proposes and describes an e-service management solution itself as a web based e-service that enables remote management of e-services in a uniform platform independent manner.

Managing Next Generation E-Services

Akhil Sahai, Vijay Machiraju, Klaus Wurster
E-services Solutions and Management Department
HP Laboratories, 1501 Page Mill Road, Palo-Alto, CA,
{asahai|vijaym|kwurster}@hpl.hp.com

***Abstract*- The ever-widening reach of Internet has led to the proliferation of e-services. These e-services are sprouting in the form of portals and e-business web sites. Some of these e-services interact amongst themselves and undergo service composition to offer other complex services. Next generation e-services would undertake dynamic service composition because of the multiple benefits this model of operation provides. In addition, these e-services have varied implementations which makes managing these e-services a challenging task. These web based e-services also operate on line and would need continuous monitoring and control. This paper proposes and describes an e-service management solution itself as a web based e-service that enables remote management of e-services in a uniform platform independent manner.**

***Index terms*—outsourced/remote management, Internet, Web, e-services, e-service management, XML**

A. INTRODUCTION

An *e-service* is a service available via the Internet that completes tasks, solves problems, or conducts transactions. An e-service drives new revenue streams or creates new efficiencies in the Internet economy. These e-services can be distinguished by the following characteristics:

1. They are accessible via the Internet at a particular URL.
2. They could be composite in nature. An e-service may depend on other e-services. This composition could be static or dynamic. The dynamic nature of composition of e-services makes their management a challenging task.
3. Their implementations could be vastly different in nature. They could be based on CORBA [5], BizTalk [4], COM, E-speak [3] or on other platforms. The diversity in their implementations makes it difficult to manage them.
4. They need to agree upon document exchange protocols to communicate and interoperate with each other.

By simplifying composition, e-services faciliatate businesses to focus on their value-added business logic while outsourcing other aspects. Such outsourcing could be done by composing e-services offered by other businesses. For example, a business that is interested in selling books could focus on that aspect while outsourcing other aspects such as shipping and payment handling. While these other aspects are very important to the overall business, outsourcing them is usually more profitable for various reasons – economies of scale, reduction in labor, specialization in one business aspect rather than multiple ones, and rapidly changing technologies to name a few. Management of services is one such aspect that requires resources (financial and human). Traditional management solutions are unsuitable for outsourcing. They need time to setup and thereafter would need continuous upgrade to keep pace with the fast changing technology behind the e-services. There is also an inherent risk of losing the heavy investment, if the solution is not meeting the e-service management needs.

E-service management involves monitoring and controlling the behavior of e-services. An e-service has usually four different stages: creation, deployment, user access/information transfer, and withdrawal. Each stage of the service needs to be managed. One way to facilitate management is to design the e-services in such a way that they can be managed at a certain level of abstraction.

This paper proposes a solution that visualizes e-service management itself as an URL based e-service (*management portal*), that communicates with the managed e-service using a management protocol. This approach enables e-businesses to outsource management of e-services to the portal. It minimizes the risk and investment of the e-services. The scalability issues (as the business grows) and continuous upgrades would be taken care of by the portal. Also the long term usefulness is guranteed by the nature of operation as e-businesses can go to other management portals in case a particular portal fails to meet their requirements.

There is thus a need to outsource [1]e-service management. However to enable this model, valid mechanisms and infrastructure must be designed. Proper mechanisms for discovery/registration of e-services, configuration and monitoring of e-services, diagnosis and correction of problems with these e-services need to be implemented.

B. Remote Management

Before exploring the details on how remote management could be accomplished, let us look at the various players and their roles (Figure 1):

Service Provider (ISP/ASP)

The service provider co-ordinates the creation/updation and maintenance of an e-service.

Client

A client could be a human user that uses an e-service. An e-service could also be client to another e-service.

E-Service

The e-service is the entity that can be accessed. Clients interact with e-services by sending documents that encapsulate requests to the e-service.

Management Service Provider

The service provider that creates, deploys and maintains the management e-service

Management e-service/E-service manager

The e-service manager manages the service instances and interacts with the e-service. Since the e-service manager is implemented as an e-service by itself, we also refer to this as the Management e-service.

Administrator

The human behind the e-service manager.

In either the outsourcing model or the simpler enterprise model, the management portal mentioned in section A could be viewed as the front end to the e-service manager. It provides customization of the management views and access to the actual management functionality.

[1] A simpler variation of the model could be an enterprise wide service manager that manages the multiple e-services being provided by the same enterprise.

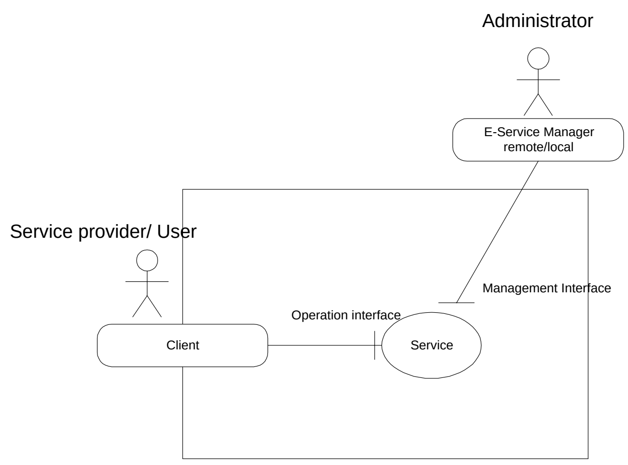

Figure 1. A managed service services interfaces with the e-service managers through the management interface

There are three important components to the remote management solution:

a. Instrumentation of e-services to provide the necessary management information and control points.
b. Management Vocabulary to communicate that information with the e-service manager.
c. E-service Manager that uses the raw management data to manage the e-service.

Each of these components is described in detail in the rest of this section.

1st. E-service Instrumentation

To enable proper functioning of the e-service manager, the managed services should provide the necessary information and control hooks to it through a set of pre-defined interfaces. *Instrumentation* is the term that is commonly used to refer to the infrastructure that has to be included in the managed service in order to support integration with a management system (Figure 2).

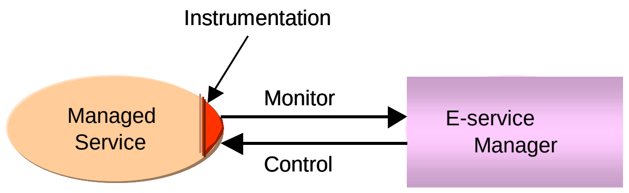

Figure 2: E-Service instrumentation

Instrumentation consists of implementation code that has to be included with the managed service (either invasively or non-invasively) in order to collect and export models and raw measurements, to signal and throw interesting events, and to implement control interfaces for use by the management system. In this section, we examine different instrumentation techniques for e-services and present some of the standard approaches used to provide end-to-end view.

1) *Instrumentation Standards*

There are several instrumentation standards such as ARM, SNMP, and CIM, that are used by network, system, and application developers to expose models, events, measurements, and control interfaces to the corresponding management systems. ARM (Application Response Measurement) is an API that defines function calls used to instrument an application for transaction monitoring. An ARM agent collects these calls and correlates them to construct the overall response time of composite transactions spanning over multiple services, as shown in Figure 3.

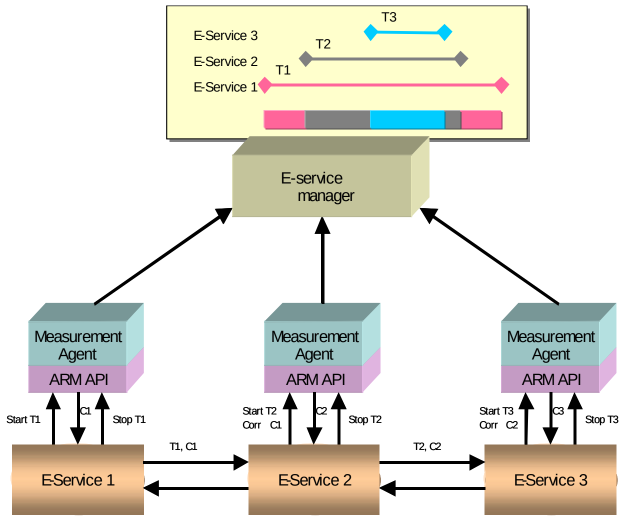

Figure 3: Using ARM instrumentation to correlate end-to-end transaction views.

SNMP (Simple Network Management Protocol) is a standard protocol used to exchange network and some system information in a simple tabular representation called MIB (Management Information Base). Each application can define its own MIB. SNMP gained wide popularity because of its simplicity. However, this simplicity prohibited defining more complex data and expressing relationships between data elements. CIM (Common Information Model) came to the rescue, with rich syntax for representing management information and relationships between managed objects.

In our designs, we used CIM to model the managed services and its workflow, and to represent transactions and metrics. We also used ARM-like mechanism to instrument the services and collect transaction measurements from all involved services, as described in Section 5.

2) *Instrumentation Options*

Instrumentation techniques fall under two main categories: invasive and non-invasive (Figure 4). Invasive techniques require adding instrumentation code within the applications that make up the service. This is usually done at design time and provides the highest level of control over the amount and kind of information provided to the management system. Non-invasive techniques are external components that are bolted in after the service is implemented, which allows instrumenting legacy applications and services. ARMing the application is an example of invasive instrumentation, while adding components that intercept transactions, extract the required information, and send it to the management system is an example of non-invasive instrumentation.

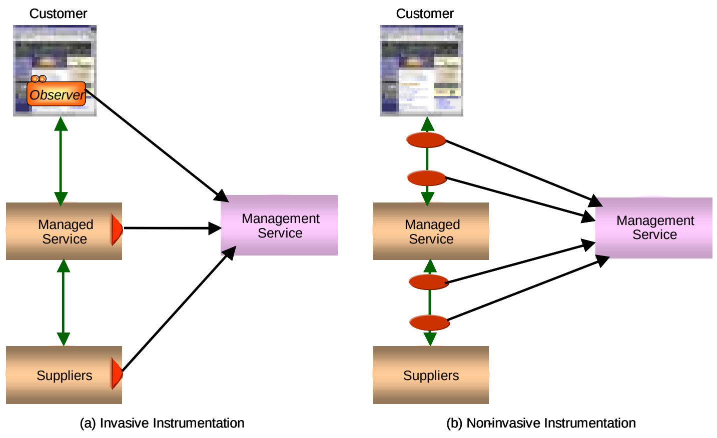

Figure 4: E-Service instrumentation options

In the invasive case, the instrumentation code could be added by service developers at design time. However, programmers tend to ignore adding these calls, as long as it is not enforced or clearly stated in the design requirements. Another way to add instrumentation is by providing development environments and tools that would automatically insert the required calls into the code. This approach may require initial setup and configuration to select the right insertion points.

The instrumented e-services are termed *well behaved services*, if they can be discovered by the e-service manager, can be configured to collect the necessary data, and can be asked to download proper instrumentations for service diagnosis and recovery in case of problem.

The management portal interacts with e-services in a *management vocabulary*. The e-service manager can provide instrumentation APIs for new e-services, that enables them to converse. For legacy applications, instrumentations in the form of interceptors may be deployed to perform this task in a non-intrusive manner.

2nd. Management Vocabulary

Once instrumented, the managed service has to send instrumentation messages to the management system. Similarly, the management system has to send control messages back to the managed service. There are two basic models of communication for such exchange of messages to occur:

Network Object Model (NOM): where the communicating parties define and agree on certain interfaces and call methods on each other.

Document Model (DM): where both parties agree on certain message formats and send documents to each other.

The NOM approach requires both the management system and managed service to define instrumentation related interfaces and their methods at construction time, which makes them strongly bound to each other. On the other hand, the DM approach allows the management system and the managed service to be loosely bound to each other and leaves more room for evolution of messages over time. In this work, we adopted the Document Model and defined certain message schemas to facilitate the exchange of instrumentation messages that are understood by both the parties. This work is based on standard agent communication languages such as KQML [13], ACL, and FLBC [11].

The first step in defining the management vocabulary is to identify the types of messages that should flow between the managed service and the E-service manager. One such classification is explained below:

Information: Informational messages about the service models, measurements collected, and events generated. Service models are used to represent the dependencies between services, permitted state transitions within the service, and workflow of the supported transactions. Examples of measurements include measurements collected whenever transactions are started and stopped at service and sub-service boundaries, availability heartbeats, contract and offer details, and lifecycle state changes. Events are the asynchronous messages sent whenever errors or important incidents occur within the service.

Requests: Requests from the managed service to provide computed metrics or to change a management policy. Requests could also be sent by the e-service manager to the managed service to improve its behavior.

Replies: These are messages that are sent in response to the requests described above.

The second step in defining the management vocabulary is to express the various categories of management information, requests, and replies as documents. We developed the *E-Management Vocabulary (EMV)* as a collection of all such terms and documents that are commonly used in the context of enterprise and outsourced e-service management.

XML [1] is used as the representation mechanism for the terms and documents. In order to define the DTDs or schemas for specifying the XML documents, we used a model-based approach. Models were designed to represent the various portions of the management information and their relationships with each other. For example, models were designed to express services, their dependecies, configuration parameters, transactions supported by services, workflow and state transitions, and the metrics generated by services. We used UML (Unified Modeling Language) to define these models. From these models, we can use XMI [10] to generate the required XML schemas and DTDs, or xmlCIM (a DMTF standard) [11] to generate CIM-compliant XML documents that could be exchanged with other CIM agents. Figure 5 illustrates this approach.

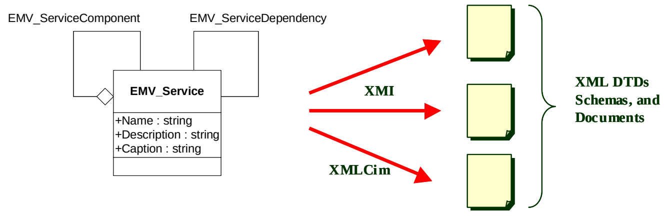

Figure 5: Generating E-Management Vocabulary from CIM models.

The messages that the e-service manager receives are either information, requests or replies which are described in the EMV as follows.

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT Message ANY>

<!ELEMENT ManagementMessage (
                      ManagementMessage.Information|
                      ManagementMessage.Request |
                      ManagementMessage.Reply
                      )>

<!ELEMENT ManagementMessage.Information (
                      ServiceType |
                      ServiceTypeComposition |
                      ServiceInstance |
                      ServiceInstanceComposition |
                       TransactionType |
                      TransactionInstance |
                      TransactionTypeComposition |
                      TransactionInstanceComposition
                      )>
<!ELEMENT ManagementMessage.Request (
                      MetricRequest
                      )>

<!ELEMENT ManagementMessage.Reply (
                      MetricReply
                      )>

<!ENTITY % ServiceTypeRef "(#PCDATA)">

<!ENTITY % ServiceInstanceRef "(#PCDATA)">

<!ENTITY % TransactionTypeRef "(#PCDATA)">

<!ENTITY % TransactionInstanceRef "(#PCDATA)">
```

The information messages contain information about the ServiceType and in case of composite services information about the ServiceTypeComposition. A service instance is an instantiated service of a particular ServiceType while a ServiceInstanceComposition is an instance of a composite service.

```
<!--------------------------------------------------->
<!–ServiceType →
<!--------------------------------------------------->

<!ELEMENT ServiceType (
            ServiceType.Name,
            ServiceType.Description?
            ServiceType.Caption?
          )>

<!ELEMENT ServiceType.Name (#PCDATA)>

<!ELEMENT ServiceType.Description (#PCDATA)>

<!ELEMENT ServiceType.Caption (#PCDATA)>

<!------------------------------------------------->
<!–ServiceTypeComposition →
<!-------------------------------------------------->

<!ELEMENT ServiceTypeComposition (
            ServiceTypeComposition.Parent,
            ServiceTypeComposition.Child
          )>

<!ELEMENT ServiceTypeComposition.Parent
%ServiceTypeRef;>

<!ELEMENT ServiceTypeComposition.Child
%ServiceTypeRef;>

<!--------------------------------------------------->
<!-- METAMODEL CLASS:  ServiceInstance →
<!--------------------------------------------------->

<!ELEMENT ServiceInstance (
            ServiceInstance.Name,
            ServiceInstance.Description?,
            ServiceInstance.Caption?,
            ServiceInstance.ServiceType?,
          )>

<!ELEMENT ServiceInstance.Name (#PCDATA)>

<!ELEMENT ServiceInstance.Description (#PCDATA)>

<!ELEMENT ServiceInstance.Caption (#PCDATA)>
```

```
<!ELEMENT ServiceInstance.ServiceType
%ServiceTypeRef;>

<!--------------------------------------------------->
<!-- ServiceInstanceComposition →
<!------------------------------------------------->

<!ELEMENT ServiceInstanceComposition (
            ServiceInstanceComposition.Parent,
            ServiceInstanceComposition.Child
          )>

<!ELEMENT ServiceInstanceComposition.Parent
%ServiceInstanceRef;>

<!ELEMENT ServiceInstanceComposition.Child
%ServiceInstanceRef;>
```

E-services undertake a set of transactions. These transactions have to be monitored and controlled. These Transactions are of certain type.

```
<!--------------------------------------------------->
<!-- METAMODEL CLASS:  TransactionType →
<!--------------------------------------------------->

<!ELEMENT TransactionType (
            TransactionType.Name,
            TransactionType.Description?,
            TransactionType.Caption?
          )>

<!ELEMENT TransactionType.Name (#PCDATA)>

<!ELEMENT TransactionType.Description (#PCDATA)>

<!ELEMENT TransactionType.Caption (#PCDATA)>

<!--------------------------------------------------->
<!-- METAMODEL CLASS:  TransactionInstance →
<!--------------------------------------------------->

<!ELEMENT TransactionInstance (
            TransactionInstance.Id,
            TransactionInstance.TransactionType?,
            TransactionInstance.ServiceInstance?,
            TransactionInstance.StartTime?,
            TransactionInstance.StopTime?,
            TransactionInstance.Success?
          )>

<!ELEMENT TransactionInstance.Id (#PCDATA)>

<!ELEMENT TransactionInstance.TransactionType
%TransactionTypeRef;>

<!ELEMENT TransactionInstance.ServiceInstance
%ServiceInstanceRef;>

<!ELEMENT TransactionInstance.StartTime (#PCDATA)>

<!ELEMENT TransactionInstance.StopTime (#PCDATA)>

<!ELEMENT TransactionInstance.Success (#PCDATA)>
```

The E-service manager also receives request and reply messages. The request messages are to obtain management information from the e-service manager and are of type MetricRequest while the replies from the e-service manager are of type MetricReply.

```
<!-------------------------------------------------->
```

```
<!-- METAMODEL CLASS:  MetricRequest -->
<!------------------------------------------------->

<!ELEMENT MetricRequest (
            MetricRequest.MetricName,
            MetricRequest.ServiceInstance,
            MetricRequest.TransactionType,
            MetricRequest.StartTime,
            MetricRequest.EndTime,
            MetricRequest.Duration
          )>

<!ELEMENT MetricRequest.MetricName (#PCDATA)>

<!ELEMENT MetricRequest.ServiceInstance
%ServiceInstanceRef;>

<!ELEMENT MetricRequest.TransactionType
%TransactionTypeRef;>

<!ELEMENT MetricRequest.StartTime (#PCDATA)>

<!ELEMENT MetricRequest.EndTime (#PCDATA)>

<!ELEMENT MetricRequest.Duration (#PCDATA)>

<!------------------------------------------------->
<!-- METAMODEL CLASS:  MetricReply  -->
<!------------------------------------------------->

<!ELEMENT MetricReply (
            MetricReply.MetricName,
            MetricReply.ServiceInstance,
            MetricReply.TransactionType,
            MetricReply.StartTime,
            MetricReply.EndTime,
            MetricReply.Duration,
            MetricReply.Value
          )>

<!ELEMENT MetricReply.MetricName (#PCDATA)>

<!ELEMENT MetricReply.ServiceInstance
%ServiceInstanceRef;>

<!ELEMENT MetricReply.TransactionType
%TransactionTypeRef;>

<!ELEMENT MetricReply.StartTime (#PCDATA)>

<!ELEMENT MetricReply.EndTime (#PCDATA)>

<!ELEMENT MetricReply.Duration (#PCDATA)>

<!ELEMENT MetricReply.Value (#PCDATA)>
```

3rd. E-Service manager

The E-service manager manages e-services remotely. An E-service manager is a logical entity and can actually be distributed physically over multiple machines. For scalability purposes multiple e-service managers can be connected to each other to handle requests.

The E-service manager is composed of a number of loosely-coupled components such as a router, a simple directory, and a set of FCAPS manager components (Figure 6). In addition, the e-service manager also has a *publish-subscribe bus* for communication between components, and a *model* for persistent storage of management information.

E-service manager components register themselves with the directory as and when these components are initialized. They also subscribe with the publish-subscribe bus for messages of their interest. All the management components have access to the model . The management components add/modify and update the model object.

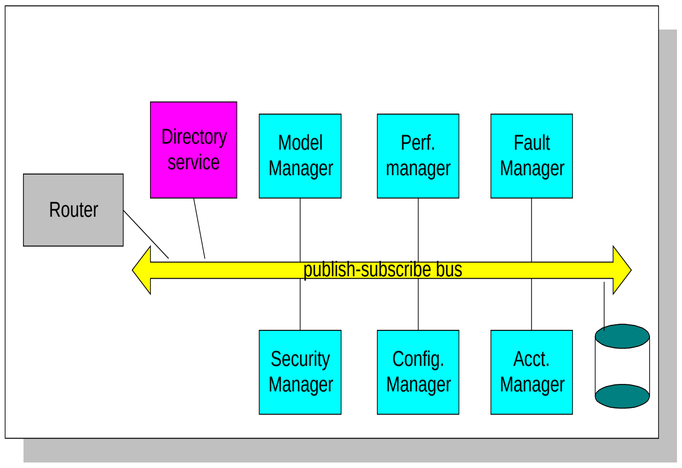

Figure 6: E-service manager

The E-service manager has a well known address (URL) in a particular protocol. All messages sent to this URL are first received by the router. It validates the messages it receives to check whether they conform to management vocabulary. These messages are of type Request, Inform or Reply. The router initates a request handler to handle a particular request. The request handler in turn routes the message either in a point-to-point manner in case of a request message or publishes it on the bus. The model manager builds the service model depending on the messages it receives. This model could be partial or complete in nature.

Protocol adapters (Figure 7) provide the E-service manager a well known address in a particular protocol. By default the E-service Manager has a http and a tcp protocol adapter. The http adapter enables e-services to connect to the e-service manager through http while tcp adapter enables e-services to connect through tcp. The former would be interesting in the outsourced model of operation while the latter is important for the enterprise mode of operation.

In addition, e-services would have varied set of implementations. These e-services could be either Biztalk, E-speak, or CORBA, Jini [2] or other Message oriented Middleware based. They would need message adapters to convert the messages sent by e-services to comply with the management vocabulary. These are termed message adapters that act as the bridge between the the e-service and the e-service manager and route valid messages to the E-service manager through the chosen protocol adapter.

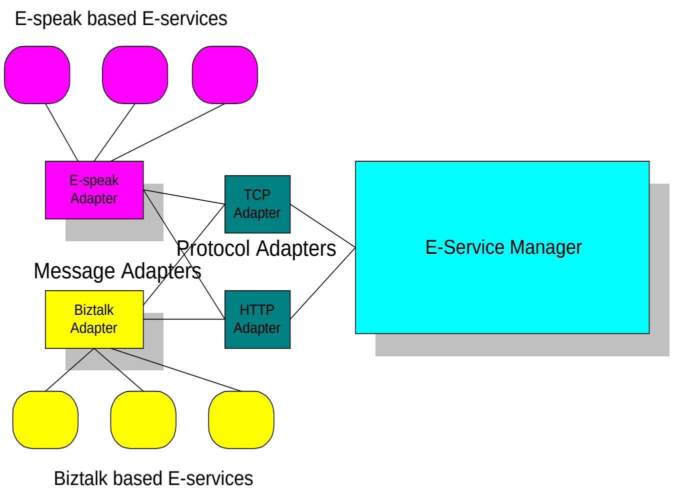

Figure 7: E-Service manager adapters

C. Sample Application

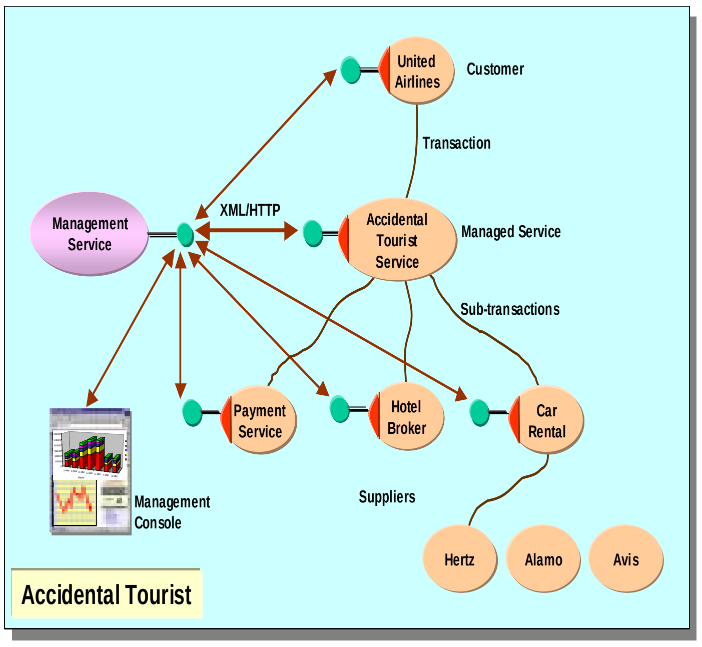

Figure 9: The Accidental tourist sample application

To demonstrate the feasibility of the proposed approach, we built a web-based e-service manager, which provides aggregate views of the end-to-end transaction response time for other services. It utilizes the EMV and communication protocols described, and provides web-based views through a management portal, which include, among other things, different remote management and information services. Figure 8 shows an overview of the managed service, which is called the “Accidental Tourist” service.

Airline companies can use the accidental tourist service whenever one of their flights is cancelled to secure accommodation and transportation to its passengers if they have to stay overnight for the next available flight. The accidental tourist service uses several external services such as a hotel broker service, car rental services (e.g., Hertz, Avis, etc.), and a payment service (e.g., Veriphone). Each of these e-services is implemented on e-speak.

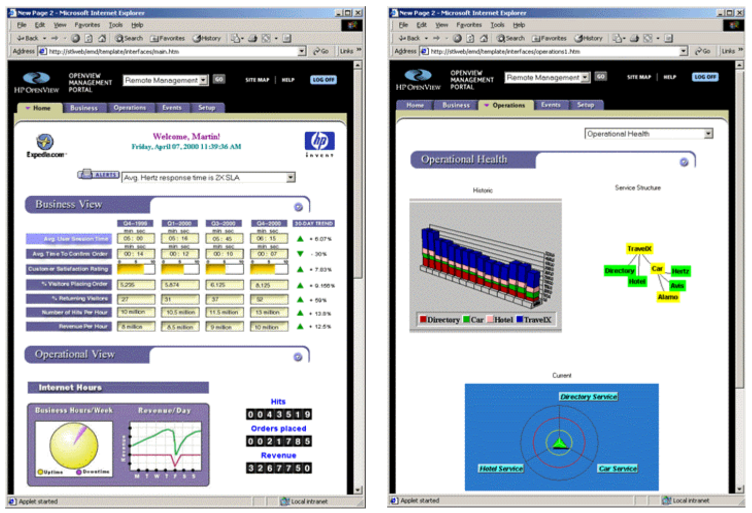

Figure 9: The web based E-service manager

The scenario starts by the managed service sending its service model to the management service followed by a configuration document specifying the required management features. The management service sets the required configuration and starts collecting low-level measurements and events in the form of XML documents from the managed service. The management console, which provides web-based customizable views to the managed service, updates its views by requesting the necessary information from the e-service manager. The e-service manager may also receive low-level measurements from some of the managed services' suppliers and consumers, based on their contractual agreements. This will provide the end-to-end view of the state of the overall service.

Figure 9 shows a screen shot of the management console, displaying various response time metrics of the accidental tourist service broken down by car rental services, hotel broker services, and by the payment service. As a further level of drilldown, one can see the average response times for each of the car rental services - Hertz, Alamo, and Avis.

D. Related Work

Several efforts have been undertaken to address various issues of service management in general. These service management solutions have however mostly targetted internet services by providing solutions for managing the web server, application servers and/or the network, middleware behind the Internet services.

Mamba [6] by Luminate.Net is a real time status and alert monitor for NT/2000, webservers, email, sql server, oracle and SAP R/3. Once installed, Mamba sends key performance information to its hosted intelligence center to automatically look for fatal problems. The system administrator is informed of problems via email. Although Mamba provides solution for managing certain platforms, it does not provide any generic solution for e-service management. The messages sent are proprietary and do not provide for cross platform e-service management.

E.M [14] by Manage.com is a similar solution. It accesses eCommerce transaction performance data from a wide range of data sources, including HTTP, HTTPS, DNS, Windows NT perfmon, SNMP and correlates top-level eCommerce service views with systems and network infrastructure performance. It however does not provide a cross platform e-service management solution based on a generic vocabulary.

Orbix [7] provides management library for instrumenting manageable CORBA services. Once instrumented with these management libraries the management information can be visualized at the Orbix Manager. This solution although satisfactory for CORBA services does not address the problem of e-service management in general.

Cisco NetSys Connectivity Service Manager [8] coupled with Service Level Management Suite provides mechanisms to establish service level policies for connectivity, reliability and security of network services. It monitors actual network configuration data and verifies the availability of key network services. The CISCO service manager monitors the networking aspects of the Internet service and does not deal with the e-service management issues as such.

E-speak service management [9] provides management libraries to instrument e-services. These libraries genrate management data which can be used for managing them. These e-speak events adhere to a proprietary format. E-speak service management although addresses the problem of managing e-speak based services, does not provide a generic remote management solution.

E. ACKNOWLEDGEMENT

We would like to thank Mohamed Dekhil, Martin Griss and Fukiko Hidano for their inputs on this paper.

F. Conclusion

E-service management is a relatively new area of research. It is also quite challenging in nature because of the federated, varied and dynamic nature of e-services. We discussed the e-service management vocabulary and e-service manager architecture that would enable remote management of e-services in a generic manner.

G. References

[1] XML at World Wide Web (WWW) Consortium. http://www.w3.org/xml

[2] K. Arnold, et. al. *The Jini^{TM} Specification*, Addison-Wesley, June 1999.

[3] Hewlett-Packard Corporation. E"Speak Architecture Specification. Version Beta2.2. December 1999. http://www.e-speak.net/library/pdfs/E-speakArch.pdf.

[4] D. Rogers, *BizTalk service framework*. Microsoft Corporation http://www.biztalk.org

[5] CORBA Specification by OMG. http://www.omg.org

[6] Mamba by luminate.Net http://www.luminate.com/company/luminate_net.html

[7] Orbix by Iona technologies http://www.iona.com

[8] Cisco Netsys Service Manager http://www.cisco.com/univercd/cc/td/doc/pcat/nesvmn.htm

[9] E-speak service management http://www.e-speak.hp.com/media/beta22/sysmgmt.pdf

[10] XML metatdata Interchange format by OMG http://www.omg.org

[11] XML CIM by Desktop management Task Force (DMTF) http://www.dmtf.org/download/spec/xmls/CIM_XML_Mapping20.htm

[12] Formal Language for Business Communications. University of Michigan http://www-personal.umich.edu/~samoore/research/flbc/

[13] Knowledge Query manipulation Language http://www.cs.umbc.edu/kqml/

[14] E.M by Manage.com http://www.manage.com