High Efficient Knowledge Extracting Platform Based on E-Delivery Pattern

Meiqun Liu, Kun Gao
Culture and Communication College
Computer Science and Information Technology College
Zhejiang Wanli University
No. 8 South Qianhu Road, Ningbo 315100
China
http://www.zwu.edu.cn

Abstract: - Generally, the existing Data Mining delivery mechanisms are based on moving data to a mining server node or moving data mining code to data resource node. This pattern is not suit for the new requirement for some situations. In this paper, we proposed a novel knowledge extraction mechanism which based on E-delivery pattern. The new mechanism can solve efficiently some problem coming from nodes having data resources but lack of computing power, or some node with computing power but no data resource and low transmission bandwidth. The kernel of the system we proposed adopts OGSA as the foundation and provides open service for systematic expansion. We also discuss the feasibility and the scalability in this paper.

Key-Words: - Software delivery, Knowledge Discovery, Knowledge Extracting, OGSA

1 Introduction

An application service provider (ASP) is a business that provides computer-based services to customers over a network. Software offered using an ASP model is also sometimes called On-demand software or software as a service (SaaS). The most limited sense of this business is that of providing access to a particular application program (such as customer relationship management) using a standard protocol such as HTTP.

The need for ASPs has evolved from the increasing costs of specialized software that have far exceeded the price range of small to medium sized businesses. As well, the growing complexities of software have led to huge costs in distributing the software to end-users. Through ASPs, the complexities and costs of such software can be cut down. In addition, the issues of upgrading have been eliminated from the end-firm by placing the onus on the ASP to maintain up-to-date services, 24 x 7 technical support, physical and electronic security and in-built support for business continuity and flexible working.

The importance of this marketplace is reflected by its size. As of early 2003, estimates of the United States market range from 1.5 to 4 billion dollars. Clients for ASP services include businesses, government organizations, non-profits, and membership organizations.

There are several forms of ASP business. These are:

- A specialist or functional ASP delivers a single application, such as credit card payment processing or timesheet services;
- A vertical market ASP delivers a solution package for a specific customer type, such as a dental practice;
- An enterprise ASP delivers broad spectrum solutions;
- A local ASP delivers small business services within a limited area.

Some analysts identify a volume ASP as a fifth type. This is basically a specialist ASP that offers a low cost packaged solution via their own website. PayPal was an instance of this type, and their volume was one way to lower the unit cost of each transaction.

In addition to these types, some large multi-line companies (such as IBM), use ASP concepts as a particular business model that supports some specific customers.

The application software resides on the vendor's system and is accessed by users through a web browser using HTML or by special purpose client software provided by the vendor. Custom client software can also interface to these systems through XML APIs. These APIs can also be used where integration with in-house systems is required.

Common features associated with ASPs include:

- ASP fully owns and operates the software application(s)
- ASP owns, operates and maintains the servers that support the software
- ASP makes information available to customers via the Internet or a "thin client"
- ASP bills on a "per-use" basis or on a monthly/annual fee

The advantages to this approach include:

- Software integration issues are eliminated from the client site
- Software costs for the application are spread over a number of clients
- Vendors can build more application experience than the in-house staff
- Key software systems are kept up to date, available, and managed for performance by experts
- Improved reliability, availability, scalability and security of internal IT systems
- A provider's service level agreement guarantees a certain level of service
- Access to product and technology experts dedicated to available products
- Reduction of internal IT costs to a predictable monthly fee.
- Redeploying IT staff and tools to focus on strategic technology projects that impact the enterprise's bottom line

Some inherent disadvantages include:

- The client must generally accept the application as provided since ASPs can only afford a customized solution for the largest clients
- The client may rely on the provider to provide a critical business function, thus limiting their control of that function and instead relying on the provider
- Changes in the ASP market may result in changes in the type or level of service available to clients
- Integration with the client's non-ASP systems may be problematic

Evaluating an Application Service Provider security when moving to an ASP infrastructure can come at a high cost, as such a firm must assess the level of risk associated with the ASP itself. Failure to properly account for such risk can lead to:

- Loss of control of corporate data
- Loss of control of corporate image
- Insufficient ASP security to counter risks
- Exposure of corporate data to other ASP customers
- Compromise of corporate data

Some other risks include failure to account for the financial future of the ASP in general, i.e. how stable a company is and if it has the resources to continue business into the foreseeable future. For these reasons Cisco Systems has developed a comprehensive evaluation guideline. This guideline includes evaluating the scope of the ASP's service, the security of the program and the ASP's maturity with regard to security awareness. Finally the guidelines indicate the importance of performing audits on the ASP with respect to:

- Port/Network service
- Application vulnerability
- ASP Personnel

Physical visits to the ASP to assess the formality of the organization will provide invaluable insight into the awareness of the firm.

2 Software as a Service

Software as a Service (SaaS, typically pronounced 'sass') is a model of software deployment where an application is hosted as a service provided to customers across the Internet. By eliminating the need to install and run the application on the customer's own computer, SaaS alleviates the customer's burden of software maintenance, ongoing operation, and support. Conversely, customers relinquish control over software versions or changing requirements; moreover, costs to use the service become a continuous expense, rather than a single expense at time of purchase. Using SaaS also can conceivably reduce that up-front expense of software purchases, through less costly, on-demand pricing. From the software vendor's standpoint, SaaS has the attraction of providing stronger protection of its intellectual property and establishing an ongoing revenue stream. The SaaS software vendor may host the application on its own web server, or this function may be handled by a third-party application service provider (ASP). This way, end users may reduce their investment on server hardware too.

The concept of "software as a service" started to circulate prior to 1999 and was considered to be "gaining acceptance in the marketplace" in Bennett et al., 1999 paper on "Service Based Software" [1].

Whilst the term "software as a service" was in common use, the CamelCase acronym "SaaS" was

allegedly not coined until several years later in a white paper called "Strategic Backgrounder: Software as a Service" [2] by the Software & Information Industry's eBusiness Division published in Feb. 2001, but written in fall of 2000 according to internal Association records.

As a term, SaaS is generally associated with business software and is typically thought of as a low-cost way for businesses to obtain the same benefits of commercially licensed, internally operated software without the associated complexity and high initial cost. Consumer-oriented web-native software is generally known as Web 2.0 and not as SaaS. Many types of software are well suited to the SaaS model, where customers may have little interest or capability in software deployment, but do have substantial computing needs. Application areas such as Customer relationship management (CRM), video conferencing, human resources, IT service management, accounting, IT security, web analytics, web content management and e-mail are some of the initial markets showing SaaS success. The distinction between SaaS and earlier applications delivered over the Internet is that SaaS solutions were developed specifically to leverage web technologies such as the browser, thereby making them web-native. The data design and architecture of SaaS applications are specifically built with a 'multi-tenant' backend, thus enabling multiple customers or users to access a shared data model. This further differentiates SaaS from client/server or 'ASP' (Application Service Provider) solutions in that SaaS providers are leveraging enormous economies of scale in the deployment, management, support and through the Software Development Lifecycle.

2.1 Key Characteristics

The key characteristics of SaaS software, according to IDC, include:[3]

- network-based access to, and management of, commercially available software
- activities that are managed from central locations rather than at each customer's site, enabling customers to access applications remotely via the Web
- application delivery that typically is closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics
- centralized feature updating, which obviates the need for downloadable patches and upgrades.
- SaaS is often used in a larger network of communicating software - either as part of a mashup or as a plugin to a platform as a service. Service oriented architecture is naturally more complex than traditional models of software deployment.

SaaS applications are generally priced on a per-user basis, sometimes with a relatively small minimum number of users and often with additional fees for extra bandwidth and storage. SaaS revenue streams to the vendor are therefore lower initially than traditional software license fees, but are also recurring, and therefore viewed as more predictable, much like maintenance fees for licensed software.

2.2 Implementation

According to Microsoft, SaaS architectures generally can be classified as belonging to one of four "maturity levels," whose key attributes are configurability, multi-tenant efficiency, and scalability.[4] Each level is distinguished from the previous one by the addition of one of those attributes:

Level 1 - Ad-Hoc/Custom: At the first level of maturity, each customer has its own customized version of the hosted application and runs its own instance of the application on the host's servers. Migrating a traditional non-networked or client-server application to this level of SaaS typically requires the least development effort and reduces operating costs by consolidating server hardware and administration.

Level 2 - Configurable: The second maturity level provides greater program flexibility through configurable metadata, so that many customers can use separate instances of the same application code. This allows the vendor to meet the different needs of each customer through detailed configuration options, while simplifying maintenance and updating of a common code base.

Level 3 - Configurable, Multi-Tenant-Efficient: The third maturity level adds multi-tenancy to the second level, so that a single program instance serves all customers. This approach enables more efficient use of server resources without any apparent difference to the end user, but ultimately is limited in its scalability.

Level 4 - Scalable, Configurable, Multi-Tenant-Efficient: At the fourth and final SaaS maturity level,

scalability is added through a multitier architecture supporting a load-balanced farm of identical application instances, running on a variable number of servers. The system's capacity can be increased or decreased to match demand by adding or removing servers, without the need for any further alteration of application software architecture.

Virtualization also may be used in SaaS architectures, either in addition to multi-tenancy, or in place of it.[5] One of the principal benefits of virtualization is that it can increase the system's capacity without additional programming. On the other hand, a considerable amount of programming may be required to construct a more efficient, multi-tenant application. Combining multi-tenancy and virtualization provides still greater flexibility to tune the system for optimal performance.[6] In addition to full operating system-level virtualization, other virtualization techniques applied to SaaS include application virtualization and virtual appliances.

Various types of software components and frameworks may be employed in the development of SaaS applications. These tools can reduce the time to market and cost of converting a traditional on-premise software product or building and deploying a new SaaS solution. Examples include components for subscription management, grid computing software, web application frameworks, and complete SaaS platform products.[7]

2.3 SaaS and SOA

Much like any other software, Software as a Service can also take advantage of Service Oriented Architecture to enable software applications to communicate with each other. Each software service can act as a Service provider, exposing its functionality to other applications via public brokers, and can also act as a Service requester, incorporating data and functionality from other services.

The traditional rationale for outsourcing of IT systems is that by applying economies of scale to the operation of applications, a service provider can offer better, cheaper, more reliable applications than companies can themselves. The use of SaaS-based applications has grown dramatically, as reported by many of the analyst firms that cover the sector. But it's only in recent years that SaaS has truly flourished. Several important changes to the way we work have made this rapid acceptance possible.

Everyone has a computer: Most information workers have access to a computer and are familiar with conventions from mouse usage to web interfaces. As a result, the learning curve for new, external applications is lower and less hand-holding by internal IT is needed.

Computing itself is a commodity: In the past, corporate mainframes were jealously guarded as strategic advantages. More recently, the applications were viewed as strategic. Today, people know it's the business processes and the data itself—customer records, workflows, and pricing information—that matters. Computing and application licenses are cost centers, and as such, they're suitable for cost reduction and outsourcing. The adoption of SaaS could also drive Internet-scale to become a commodity.[8]

Insourcing IT systems requires expensive overhead including salaries, health care, liability and physical building space.

Applications are standardized: With some notable, industry-specific exceptions, most people spend most of their time using standardized applications. An expense reporting page, an applicant screening tool, a spreadsheet, or an e-mail system are all sufficiently ubiquitous and well understood that most users can switch from one system to another easily. This is evident from the number of web-based calendaring, spreadsheet, and e-mail systems that have emerged in recent years.

Parametric applications are usable: In older applications, the only way to change a workflow was to modify the code. But in more recent applications, particularly web-based ones, significantly new applications can be created from parameters and macros. This allows organizations to create many different kinds of business logic atop a common application platform. Many SaaS providers allow a wide range of customization within a basic set of functions.

A specialized software provider can target global markets: A company that made software for human resource management at boutique hotels might once have had a hard time finding enough of a market to sell its applications. But a hosted application can instantly reach the entire market, making specialization within a vertical not only possible, but preferable. This in turn means that SaaS providers can often deliver products that meet their markets' needs more closely than traditional "shrinkwrap" vendors could.

Web systems are reliable enough: Despite sporadic outages and slow-downs, most people are willing to use the public Internet, the Hypertext Transfer Protocol and the TCP/IP stack to deliver business functions to end users.

Security is sufficiently well trusted and transparent: With the broad adoption of SSL, organizations have a way of reaching their applications without the complexity and burden of end-user configurations or VPNs.

Availability of enablement technology: According to IDC, organizations developing enablement technology that allow other vendors to quickly build SaaS applications will be important in driving adoption. Because of SaaS' relative infancy, many companies have either built enablement tools or platforms or are in the process of engineering enablement tools or platforms. A Saugatuck study shows that the industry will most likely converge to three or four enablers that will act as SaaS Integration Platforms (SIPs).[9]

Wide Area Network's bandwidth has grown drastically following Moore's Law (more than 100% increase each 24 months) and is about to reach slow local networks bandwidths. Added to network quality of service improvement this has driven people and companies to trustfully access remote locations and applications with low latencies and acceptable speeds.

2.4 Limiting Factors

Widespread implementation of SaaS requires that the services be well defined. That can achieve an economy of scale and the capacity to balance supply and demand. This requires areas of IT that are ubiquitous and commodity-like. SaaS is therefore not suitable for innovative or highly specialized niche systems, though SaaS may be used to provide one or more components in such systems.

As with manufacturing, a lack of substitutability and second sourcing options with any commodity creates a strategic weakness for any customer in terms of security, competition and pricing. Various forms of this weakness, such as 'vendor lock-in', are often cited as a barrier to adoption of SaaS as the current industry lacks portability and interoperability between vendors. This means that to change from one vendor to another will take a considerable amount of effort and time. This situation is resolvable by the introduction of open sourced standards and the development of markets based upon such standards.[10]

Whilst the severe lack of substitutability is unresolved, many vendors counter the concerns over potential security and operational risk with the argument that the professionals operating SaaS applications may have much better security and redundancy tools available to them.

Furthermore the concern that SaaS applications pose some difficulty for businesses that need extensive customization is countered with the claim that many vendors have made progress with both customization and publication of their programming interfaces. It should be noted that customization will reduce substitutability and given that SaaS covers commodity-like activities, the strategic benefit of customization is highly dubious.

In addition to this, the availability of open source applications, inexpensive hardware and low cost bandwidth combine to offer compelling economic reasons for businesses to operate their own software applications, particularly as open source solutions have become higher quality and easier to install.

Users of SaaS must be able to trust the provider of the service, particularly if the application stores the user's data. The provider needs to be trusted with both the intention and the ability to safeguard this information.

3 E-service Delivery Pattern for Data Mining

Data mining is emerging as a suitable and commercially viable application for delivery as an e-service. This is evident from the growing number of commercial data mining application service providers (ASPs). In this section we review the operation of commercial data mining ASPs.

In the commercial domain, the modus operandi for data mining ASPs is illustrated in figure 1. A client organization has a single service provider who meets all the data mining needs of a client. The client is well aware of the capabilities of the service provider and there are predefined agreements regarding quality of service, cost and protocols for requesting services. The service provider hosts one or more data mining systems that support a specified number of mining algorithms. The interaction protocol for this model is as follows:

1. Client requests a service using a well-defined instruction set from the service provider.
2. The service provider maps the request to the functionality of the different data mining

systems that are hosted to determine the most appropriate one.

3. The "suitable" data mining system processes the task and the results are given to the client in a previously arranged format. The access to the results is typically done through a secure web interface or results can be sent as files using email or FTP.

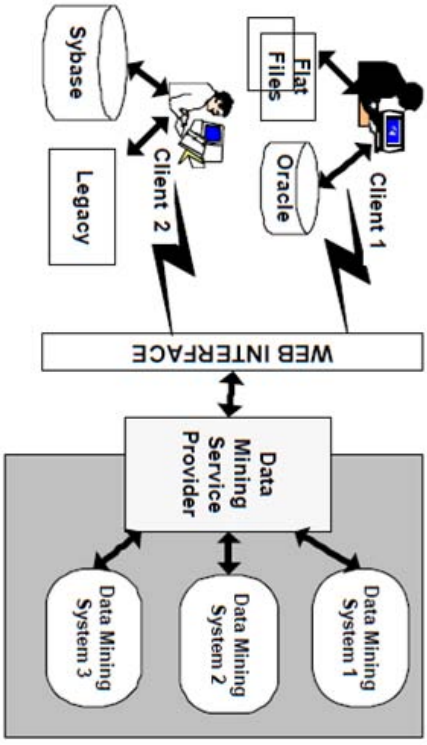

Fig.1 Operation of Commercial Data Mining Service Providers

The developments in e-services indicate the emergence of platforms to support virtual communities of e-services on the Internet. We have reviewed the operation of data mining ASPs from which it can be seen that the predominant approach adopted by them is not truly representative of the dynamic and ad-hoc vision of discovery and interaction between organisations supported by environments and standards such as E-Speak™ and UDDI. In order to illustrate the operation of a virtual community of data mining e-services, we present a model shown in figure 2. This model is characterized by clients being able to request data mining services from several service providers who host one or more data mining systems.

This approach provides a higher level of flexibility for the client and represents the establishment of an open, virtual community of data mining e-services. The model is consistent with the operation of e-services environments such as E-Speak™, which provides support infrastructure for discovery and interactions between clients and data mining service providers. The interaction protocol for this model is as follows:

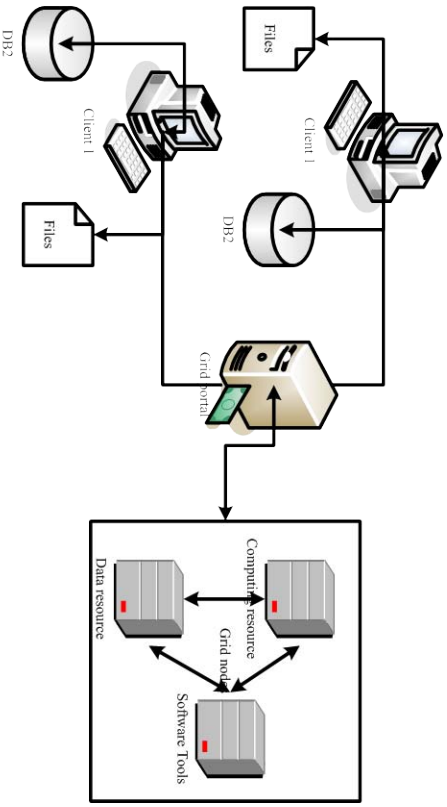

Fig. 2 E-service Delivery Pattern

i. The client requests a service by providing a task specification.

ii. The Virtual Community Manager (VCM) component broadcasts the client's requests to the data mining service providers that are registered with it. The VCM maintains information about each data mining service provider such as the name, address, contact information, data mining systems hosted, algorithms, architectures and functionality supported by those systems and the available computational resources.

iii. The data mining service providers evaluate the requested task against the capabilities and functionality of the data mining systems that they host.

iv. If they can meet the needs of the requested task, the data mining service providers respond by presenting an estimate of the cost, possible time frame for completion of the task and liabilities for not meeting the target response time. This information is presented to the community manager in a specific, structured format.

v. The VCM can present the responses it receives along with the information that it already maintains about the respective service providers to the client. Alternatively, in more sophisticated environments the VCM can perform matching of client preferences with the capabilities of the service providers, rank the service providers on that basis and present this to the client. There is also scope for automated negotiation to be incorporated into this stage of the interaction protocol.

vi. The client decides which service provider it deems most appropriate and informs the VCM and the chosen service provider.

vii. The service provider gives the client a legal document/contract, which reflects the commitment to maintain confidentiality of the data that is mined and the consequent knowledge that is produced.

viii. The client is then required to provide a security deposit in the form of a credit card number to the VCM. The actual payment is made on completion of the task and delivery of results to the client.

ix. The client and the data mining service provider exchange information regarding the transfer of data, passwords to access systems and the mode of transfer of results.

x. The data mining service provider processes the task, provides the results to the client (in the agreed format and method) and informs the VCM of task completion.

xi. The client acknowledges the completion of the task to the VCM and the payment is made to the service provider.

The above model is a hypothetical illustration of the functioning of a virtual community of data mining e-services. Current research and development in e-services enable this model by catering for the requisite functionality of the coordinating entity in the model, namely, the VCM. This includes providing the infrastructure for registering and discovering e-services [11], [12], [13] ,[14],[15], specifying protocols for interaction (Web Services Conversation Language (WSCL) [16][17] and accessing services (Web Services Description Language (WSDL) [18]). There is also an emerging recognition for the need to have strategies to support ranking and negotiation in such environments [19], [20], [21], [22]. This paper does not intend to focus on these issues in the model, which are adequately addressed by current research and development in industry, academia and standards organizations [23]. This paper addresses the questions that are specific to the operation of data mining e-services in environments characterized by the model presented in figure 2. The research presented in this paper focuses on the interactions and infrastructure needs of data mining e-services. Interactions refer to methods by which the entities in the model, namely the clients, the data mining e-service providers and the VCM communicate with one another to perform a data mining transaction. Infrastructure refers to the data mining systems that are hosted by the service provider and operate in an environment that is characterised by the provision of e-services on the Internet. We first review research in the area of supporting interactions in virtual communities of e-services including work in the data mining domain. This is followed by a discussion on the infrastructure issues that arise in the context of data mining systems operating efficiently in such an environment. We survey related work in distributed data mining in order to analyse the strengths and limitations of existing models to meet the specific needs of a virtual community of data mining e-services.

E-services technologies discussed above provide the underlying framework to enable the vision of e-services on the Internet. That notwithstanding, the interaction model and data mining infrastructure that can be tailored to meet the constraints and needs of a virtual community are fundamental pre-requisites to enable the operation and deployment of data mining e-services in such an environment. Our discussion focuses on mechanisms to support interactions that exist in various e-service frameworks and standards. We also evaluate data mining query and specification languages to establish their applicability and suitability to meet the specific interaction requirements of a virtual community of data mining e-services.

4 Interaction Requirements of E-Service Delivery pattern for Data Mining

E-services standards and frameworks facilitate interactions by providing mechanisms for specifying the protocol in terms of the messages and the order in which they need to be exchanged. These messages serve the following requirements:

- Specify the information content that needs to be exchanged between the providers and consumers of an e-service.
- Capture the requirements of clients and the capabilities of service providers.
- Specify how clients and service providers mutually interact with each other.

- Form the basis for discovering, matching, ranking and negotiation in e-services.

There are primarily two areas of study that have the potential to facilitate exchange of information between clients and service providers in a virtual community of data mining e-services, namely, the e-services environments themselves and data mining languages. In this section we review how e-services standards and frameworks facilitate interaction between clients and service providers and examine data mining languages to evaluate their applicability for supporting interactions.

4.1 Supports for Interactions in E-Services Environments

Models for supporting interactions vary among the different e-services standards and frameworks. However, the general principle is that they all provide mechanisms for e-services to specify their respective interaction protocols. In this section we review interaction support provided by the following e-services standards:

- Conversation Description Language (CDL)
- Web Services Conversation Language (WSCL)
- Universal Description, Discovery and Integration (UDDI)
- eb-XML (e-business XML)
- Web Services Description Language (WSDL)

Conversation Description Language (CDL) and its improvement Web Services Conversation Language (WSCL) were developed at Hewlett Packard to provide a mechanism to model the interactions a service supports (e.g. that a client must log in to the system prior to requesting a catalogue of services). The WSCL specification is an XML schema and the full specification is available at http://xml.coverpages.org/wscl.html. It contains the following three components:

- Document Type Descriptions, which specify the type of XML documents that the service can accept and transmit.
- Interactions, which specify the states of the conversation as a series of document exchanges between the participants. The four types of interactions supported in CDL include: Send, Receive, ReceiveSend and SendReceive.
- Transitions, which specify the ordering of the exchanges in terms of source interactions, destination interactions and the document types that triggers the transition.

WSCL enables specification of a set of rich protocols necessary to support complex interactions. It requires an e-service to provide a set of XML documents that represent messages that have to be exchanged and the order in which these messages must be exchanged. This exchange is termed a conversation and the content of the messages constitutes the information that is exchanged.

Interactions in eb-XML are supported by the Collaboration Agreement Protocol [24] which is a mechanism for describing a mutually agreed upon business arrangement and modelling of business processes. The specification for the interaction is derived by collecting the following information from each participant:

- Business Profiles that describe the clients and service providers.
- Business Processes that describe the roles, responsibilities and relationships with collaborators.
- Business Service Interfaces that specify the software interfaces to the applications that are hosted.
- Business Messages that have to be exchanged between the clients and service providers.

The eb-XML technical architecture provides the infrastructure to support the specification, collection and maintenance of this information in a component called the Registry.

In UDDI, the interactions are specified using the tModel or Technical Model component of the UDDI schema. The tModel element in the UDDI schema contains meta-data about the technical specification of the e-service. Organizations registering their services with the UDDI registry are required to provide information on how to interact with the service in a technical specification. The tModel has a reference to the technical specification (which can be a WSDL document or a WSCL document) information specified in the tModel element includes: a unique key, name, an optional description and a URL for the technical specification.

Web Services Description Language (WSDL) does not have built in support for conversations, but supports interactions because it is an XML language for describing the interfaces of e-services. A WSDL document describes the operations provided by an e-service as XML document exchanges. The messages element in a WSDL document specifies the data that needs to be exchanged between the clients and the providers of the e-service.

The above discussion on support for interactions between clients and service providers indicates that

interactions are a vital aspect of e-services and that the major e-services frameworks and standards provide mechanisms for services to describe how clients should interact with and access the service. XML messages are the lingua franca for exchanging information between clients and service providers. The service providers are required to provide the e-services environment with detailed specifications of the structure and content of the messages and the order in which they are exchanged. The concept of service specific messages with their information content tailored for the service in question is fostered by e-services environments, be it in terms of a vocabulary or a tModel specification or a Collaboration Agreement Protocol. This need for specifying the information content that has to be exchanged between clients and service providers, and the necessary protocol to support coherent interaction in e-services is leading to active research in ontologies and markup languages for vertical domains. The current focus of interaction is on discovery and access since e-services environments primarily provide support for these processes. However, developments in areas such as negotiation and ranking necessitate support for richer and more complex interactions. In the context of data mining e-services, this leads to the question of research in specifying the messages necessary to support interactions between consumers and providers of data mining e-services. We now present a survey of languages for data mining and assess their applicability to support interactions in data mining e-services.

5 Conclusion

In this paper we have discussed e-services standards and frameworks and the current operation of data mining service providers. The current modus operandi of Internet-based data mining service providers does not represent a virtual community of services. It does not provide clients with benefits of a wide choice of data mining e-services and does not cater for selection of the most appropriate and cost effective service provider. However, the developments in the e-services domain present an opportunity for realisation of a virtual community of data mining e-services by providing the architectural framework necessary for such a community. We have presented a model for a virtual community of e-services that is consistent with the operation of e-services environments. The operation of a virtual community of data mining e-services requires support for interactions between clients and service providers and infrastructure specific to e-services environments. Hence this will be explored further in this paper.

References:

[1] K. Bennett , P. Layzell , D. Budgen , P. Brereton , L. Macaulay , M. Munro, Service-based software: the future for flexible software, Proceedings of the Seventh Asia-Pacific Software Engineering Conference, p.214, December 05-08, 2000

[2] Strategic Backgrounder: Software as a Service. www.siia.net/estore/ssb-01.pdf

[3] a b Traudt, Erin; Amy Konary, June 2005, Software as a Service Taxonomy and Research Guide.

[4] Frederick Chong and Gianpaolo Carraro, Architecture strategies for catching the long tail, April 2006, http:// msdn2.microsoft.com/en-us/library/aa479069.aspx

[5] Wainewright, Phil, October 2007, "Workstream prefers virtualization to multi-tenancy.

[6] Chong, Fred, October 2006, Multi-tenancy and Virtualization

[7] Schuller, Sinclair, March 2007, Repealing the SaaS Tax.

[8] http://www.saasblogs.com/2006/09/26/scale-as-a-commodity-2/ SaaSBlogs: Scale as a Commodity

[9] SaaS 2.0: Saugatuck Study Shows Rapid SaaS Evolution to Business Platforms, April 2006.

[10] Breaking down the saga of vendor lock-in, http://wolfpaas.blogspot.com/2008/10/vendor-lockin-are-you-kidding.html

[11] UDDI Version 2.0 Data Structure Reference. Available Online: http://www.uddi.org/DataStructure-V2.00-Open-2001-0608.pdf

[12] Kobielus, J, G., (2001), "BizTalk: Implementing Business-to-Business E-Commerce", Prentice-Hall PTR, New Jersey, USA.

[13] Graupner, S., Kim, W., Lenkov, D., Sahai, A., (2001), "E-Speak-an Enabling Infrastructure for Web-based E-Services", International Conference on Advances in Infrastructure, Electronic Business, Science, and Education on the Internet (SSGRR-2000), July 31-Aug 06, L'Aqila, Italy, ISBN:88-85280-52-8. Proceedings Available Online: http://www.ssgrr.it/en/ssgrr2000/proceedings.htm

[14] eb-XML Technical Architecture Specification V1.0.4 Available Online: http://www.ebxml.org/specs/ebTA.pdf
[15] eb-XML Business Process Specification Schema V1.0.1 Available Online: http://www.ebxml.org/specs/ebBPSS.pdf
[16] Kuno, H., Lemon, M., Karp, A., and Beringer, D., (2001), "Conversations + Interfaces = BusinessLogic", Hewlett-Packard Labs Technical Report HPL-2001-127, Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-127.html
[17] Kuno, H., Lemon, M., and Beringer, D., (2001), "Using CDL in a UDDI Registry 1.0", UDDI Working Draft Best Practices Document, Hewlett-Packard Labs Technical Report HPL-2001-72, Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-72.html
[18] Web Services Description Language (WSDL) Specification. Available Online: http://msdn.microsoft.com/xml/general/wsdl.asp
[19] Bartolini, C., and Priest, C., (2001), "A Framework for Automated Negotiation", Hewlett-Packard Labs Technical Report HPL-2001-96. Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-90.html
[20] Priest, C., Byde, A., Bartolini, C., Piccinelli, G., (2001), "Towards Agent-based Service Composition through Negotiation in Multiple Auctions", Hewlett-Packard Labs Technical Report HPL-2001-96, Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-90.html
[21] Tewari, G., and Maes, P., (2000), "Design and Implementation of an Agent-based Intermediary Infrastructure for Electronic Markets", Proceedings of the Second ACM Conference on Electronic Commerce 2000, Oct 17-20, Minneapolis, USA, pp. 86-94.
[22] Tewari, G., and Maes, P., (2001), "A Generalized Platform for the Specification, Valuation, and Brokering of Heterogeneous Resources in Electronic Markets", E-Commerce Agents, (eds) J. Liu and Y. Ye, Lecture Notes in Artificial Intelligence (LNAI) 2033, Springer Verlag, pp. 7-24.
[23] Casati, F., and Shan, M., (2001), "Models and Languages for Describing and Discovering E-Services", ACM SIGMOD Record, Vol. 30, Issue 2, June. Presentation Slides of the Tutorial Delivered at the ACM SIGMOD Conference, Santa Barbara, USA, May are Available Online: http://www.hpl.hp.com/personal/Fabio_Casati/publications.html#tutorials
[24] eb-XML Technical Architecture Specification V1.0.4 Available Online: http://www.ebxml.org/specs/ebTA.pdf