WISE - Building Simple Intelligence into Web Services

Soe-Tsyr Yuan[1] and Kwei-Jay Lin[2]

[1]*Dept. of Management Information System*
National Chengchi University, Taipei, Taiwan

[2]*Dept. of EECS*
University of California, Irvine, CA 92697, USA

ABSTRACT

Web Services are self contained and self described modular applications that can be published, discovered and employed on the Web. Many standard protocols supporting web services have been adopted and more are being proposed. In this paper, we study the issues on providing intelligent web services. We propose the enhancement of web service functionalities by deploying software agents on both server side and/or client side. Our goal of designing the WISE web service architecture is to provide a working middle ground between the current web service standards and the Semantic Web architecture. The WISE software agent architecture for web services is presented. We discuss the design issues of WISE. We also present the QoS management protocol and algorithm that can be used by WISE servers.

1. INTRODUCTION

With the steady expansion of global Internet connection and servers, and the availability of rich, resourceful content, World Wide Web has grown from a small research-oriented network into a ubiquitous connection for businesses of all sizes. Companies have used Internet to attract and identify new customers, to explore new product ideas, and to expand global new market. Moreover, in the last few years, Internet has become an invaluable B2B infrastructure to streamline supply chains, to lower business cost, and to provide real-time information and feedback to suppliers, vendors and consumers.

In the past, web servers were used mainly to supply information, either static or dynamic. With companies adopting more and more IT systems in their business processes, Internet servers are becoming service-oriented, providing automated responses to requests and queries from both human users and business machines. Companies have allowed their partners and customers to access on-line information from their web servers using company-specific access mechanisms built on top of standard Internet protocols. The ease of use and the efficiency of such mechanisms make services from web connections an instant attraction.

However, despite the success of service-oriented web computing for some, a major impediment for its wide adoption and deployment for others is the non-standard access mechanisms and protocols used by most of today's web servers. Different e-commerce web sites and web servers have utilized different mechanisms to connect buyers, suppliers, marketplaces, and partners. In working with different customers, a company may need to have many communication modules each to connect to a specific customer site using a specific protocol. For companies with complex business processes it will be close to impossible to work with all of the different protocols when trying to serve a wide range of vendors and customers.

To solve the problem, standard *web services* protocols have been proposed to describe, to discover, to connect, and to integrate services provided by different companies' web servers. Under the proposed framework, web services are supposed to be "self-contained, modular business applications that have open, Internet connected, standard based interfaces." Standards such as SOAP, WSDL and UDDI have been defined to provide the foundation for an open web service framework. The goal of the web services framework is to solve the problems with proprietary business protocols but still provide enough flexibility for different industry domains so that each web service site can deliver its service correctly and efficiently when requested.

Although current web services standards provide a good start, there is still much to be explored to make web services powerful and flexible. In this paper, we study two such issues for web services: *service delivery* and *QoS*. Service delivery, that demands an effective and efficient performance, is critical for web applications and platforms that must adapt to dynamic user demands and mobile target environment when delivering services and information. QoS, that defines service quality such as latency, availability, timeliness and reliability, is important for many web applications that process real-time information, have multimedia content, or must have a guaranteed level of service. For many web services, service delivery and QoS are as critical as service's functional data result.

1.1 WISE Web Services

Instead of refining the current web services standards (such as the Semantic Web initiative [8]), which will need a big effort to go through the standard approval process, we propose to utilize *software agents* to enhance current web services. Using agents allow us to monitor and to modify the behavior of a web service module without making changes to the module itself. Software agents have prevailed throughout the years. They were explained as *software things that know how to do things that you could probably do yourself if you had the time* [9]. They embody certain characteristics such as autonomy, proactivity, adaptation, social ability, *etc.* [7]. Accordingly, software agents, by nature, come into being as a technique to (partially) realize the automation of assessment and management of customer demands so as to select the service with the best achievable quality and make the most effective web service delivery.

In this research, we investigate the deployment of simple web service agents that are easy to implement and efficient to execute. As many have suggested, systems that are implemented with the "keep it simple and stupid" (KISS) principle actually have a much better chance of success than anything that is supposed to be powerful but complex. In this paper, we present the design of *WISE (Working, Intelligent, yet Simple E-commerce)* web services. WISE web servers are supposed to work under standard web service protocols, yet have simple service intelligence and friendly user convenience built in. WISE agents rely on many simple strategies to make their host web services perform better without a major effort or perception from users (or target automated applications). Our goal of designing WISE web services is to provide a middle ground between the current web service standards and the Semantic Web initiative. Although Semantic Web [8] provides a good plan for fully intelligent web (in terms of automated web service discovery, execution, composition and interoperation), it will still take much effort and time to make it a practical reality. On the other hand, by building WISE servers using the current web service interface protocol, we can easily build a web of WISE servers. In other words, WISE is to be a working model today, building lightweight intelligence into today's web services. Efficiency and Effectiveness will be our main concern rather than powerfulness and completeness. In the future, WISE may further play as the middleware utilizing the automatic intelligence gathered from the web of services by the Web Intelligence initiative.

The rest of this paper is organized as follows. Section 2 reviews the requirements on web service delivery and QoS management. Section 3 discusses the possibilities on how software agents may be used to make web services more intelligent and easy to use. We present the WISE system architecture in Section 4. The implementation of the WISE infrastructure and QoS management protocol are briefly shown in Section 5. The paper is concluded in Section 6.

2. Requirements of Service Delivery and QoS

2.1 Issues on Service Delivery

Plain web service delivery simply transmits the specific service outcomes the service intends to accomplish. The service outcomes can be as simple as a few returned three-digit area codes for a particular state or as complex as searching and manipulating information on the web.

However, the new vision of the web is to provide a platform for service sharing, interoperation, and delivery in intelligent ways. Accordingly, plain web service delivery should be upgraded so as to exhibit certain level of intelligence. The fundamental issues involved in a service delivery strategy are as follows: (excluding the decision of a service choice, that is, *a designated service is presumed for conducting the service delivery*):

- *Demand assessment*: In assessing the demand for services the following questions need to be answered: *for whom, where, when, for how long, at what level, at what cost, etc.*
- *Demand monitoring and analysis*: Assessment and management of demand is a continuous process. Subsequent analysis of demand may reveal that current usage has altered. Demand should therefore be monitored regularly to avoid unpleasant surprises and to assure certain performance of the service.
- *Demand Communication*: Means of customer communication with the service should be effortless and without trouble.
- *Demand prioritization*: Resources available to the services are finite. However, there might be demand for services in excess of the resources required to provide them. Therefore, it is imperative to prioritize service delivery for maximizing benefits to the set of service requestors as a whole.

Plain web service delivery does not consider assessment and management of customer demands and may result in ineffective service delivery. To provide an intelligent platform for service sharing and interoperation, it is crucial to find methods to (partially) achieve the automation of assessment and management of customer demands. On the other hand, more and more people will be accessing the web from mobile phones, personal digital assistants (PDAs), auto PCs, and a variety of information appliances other than desktop PCs. Therefore, web service delivery has to consider physical limitations of these Internet-enabled mobile devices.

2.2 QoS for Web Services

Future business systems require a seamless integration of business processes, business applications, business intelligence, and web services over the Internet. Delivering QoS for most web services is a critical and significant challenge because of the dynamic and unpredictable nature of business applications and Internet traffic. Business applications with very different characteristics and requirements compete for the resources used to provide the web services. Without a careful management of web service QoS, critical business applications may suffer detrimental performance degradation, and resulting in customer dissatisfaction or financial losses.

The area of QoS management covers a wide range of techniques that match the needs of service requestors with those of the service provider's. QoS has been a major area of study in computer networking [5, 13], real-time computing [10, 17], system middleware [1, 3]. For web services, QoS guarantee and enhancement have started to receive some attention [11, 12]. However, to our knowledge, no commonly agreed framework and major performance result has been reported yet.

In our study, we consider the following quality attributes as part of the web service parameters.

1. **Response time**: the amount of time to get a service request responded at the client side. This includes the total time for service and round-trip communication.
2. **Service cost**: assuming that some or future web services will be fee-based, cost is the dollar amount for each unit of service. A web service may be priced differently depending on the quality of the service requested.
3. **Network bandwidth**: the network bandwidth required to receive the service. This is especially important for services with multimedia content such as video or large graphics. The bandwidth attribute will also be important for web service brokers to decide if a service should be invoked if the client is using a low bandwidth network such as some of today's wireless connection.
4. **Service availability**: the probability that the service is available. This only measures the server availability in terms of responding to a request, not the result quality.
5. **Result dependability**: the quality of the result produced by the web service. The dependability index is a number between 0 and 1, with 0 being unacceptable and 1 being a perfect result. This index may be generated from a complex formula combing several other factors such as timeliness of data, precision of calculation, etc.

Since our goal is to build simple intelligence into web services, the QoS attributes to be considered must be easy to understand and to measure. Except for result dependability, all of the above attributes can be easily collected by a software agent on a system without any user intervention or input. For example, before and after each connection and invocation of a web service, a software agent can automatically measure the response time, the service cost, the bandwidth used, and the number of connection attempts before the service is successfully fulfilled. The problem with the dependability measurement is that its rating is quite subjective. It is up to the user to make a judgment on how good the service has just been received.

3. Software Agents and Web Services

In this section, we present different ways software agents may be deployed in WISE. We also provide references to some of our projects implementing those capabilities.

3.1 Agent-Based Demand Assessment

The demand of a customer for a designated service unfolds as a combination of the answers to the questions as for whom, where, when, for how long, at what level, at what cost, *etc.* This combination of information is also denoted as a *context* according to the notion of context defined by Dey, *et. al.* [6]. Agents can be used to overcome the inherent input/output limitations of mobile devices and enable certain degrees of demand-assessment automation. In other words, agents embody the capabilities of understanding the context within which their users operate, such as the *environment-centric contextual attributes* (their locations, nearby persons or objects) and the *human-centric contextual attributes* (the activities they are engaged in, who their friends and colleagues are, and interaction histories and preferences). That is, they are agents with context awareness. Subsequently, agents invoke[1] the designated services with the contextualized demands.

An agent-based assessment of contextualized demands has been implemented in the Voice-Based Mobile Community Web Services (CAVBMC) project [14]. CAVBMC is a highly interactive voice-based mobile community server. CAVBMC embodies two services: *TakeService* and *OfferService*. *TakeService* enables a user to acquire from CAVBMC relevant voice information, while *OfferService* allows a user to render to CAVBMC his/her valuable experience that is expressed in audio format and location-tagged. A server with multiple software agents residing on top of the CAVBMC server manages and exploits the contexts of all of its customers. The multiagent server records and analyzes users' locations, past interactions, and other users' interactions at the same locations in order to enable contextualized use of the web services.

3.2 Agent-Based Demand Monitoring and Analysis

Agents may also be used to monitor the progression of the context and discover the possible patterns of context evolution so as to generate additional demands that are also of interest to the customer (in addition to the demand with respect to the current context). We have built the Mobile Advertising Web Services (MALCR) [22] which provides

[1] This invocation won't take effect unless it is confirmed by the customer. In other words, the customer still retains the control of service execution.

mobile advertising web services for the advertisement of commercial/non-commercial activities in order to complement the Internet and interactive television advertising and made it possible for advertisers to create tailor-made campaigns targeting users according to where they are, their needs of the moment and the devices they are using. MALCR has two services: *PullService* and *PushService*. These services generate the recommendations of the Mobile Ads that are relevant to the interests of the mobile users and the locations where the mobile users requested *PullService* or the locations where the mobile users last made use of their mobile devices so as to avoid the expensive location tracking of the users during the use of *PushService*.

A multiagent server lodging on top of the MALCR server manages and employs the contexts of all of the customers. The multiagent server records, monitors, and analyzes the historical interactions of the customers in order to discover the possible customers needs that can be further served with *PushService*.

3.3 Agent-Based Demand Communication

Agents can embody the capabilities of (multi-modal) speech processing for facilitating the customer-to-service information-exchange interactions. Our Voice-Based Knowledge Management Web Services (VBKM) [18] project has been built to facilitate knowledge acquisition and knowledge queries by speech so as to enable the employees to act as mobile workforce. VBKM attains a high quality of knowledge acquisitions and queries based on the structures of the speech documents. VBKM involves the technologies such as wireless communication, speech recognition for English, ontology, text mining, and knowledge base.

VBKM has two services: *QueryService* and *AcqusitionService*. *QueryService* enables a mobile worker to acquire from the organization memory the most relevant knowledge to a given oral query, while *AcquisitionService* allows a mobile worker to render to the organization memory his/her oral working experience that would subsequently be classified into a cluster of similar documents. Upon the request of these web services, an agent residing at the mobile device of a mobile worker processes the oral inputs and manages the context of the worker in order to enable effective service communication and delivery.

3.4 Agent-Based QoS Negotiation and Prioritization

Agents enable service resource allocation among service requests from many customers in order to prioritize service deliveries so as to maximize the benefit among service requestors as a whole. The allocation method can be the likes of market-based task allocation, contract-net based allocation, *etc*. Web service QoS concerns the allocations among web service providers (and thus their resources) and service requestors. Automated negotiation [15], which often is deployed to automate task/resource allocation in electronic commerce, then can be employed to automate web service QoS negotiation. Automated negotiation denotes the interactions among the agents representing service providers and service consumers based on communication for the purpose of coming to an agreement for distributed conflict resolution or distributed decision making.

In WISE, we concentrate only on simple QoS strategies as we assume that most web services to be deployed will have very simple cost structure or even can be used for free. The decision of which server to use or what service level to request is probably based more on context information such as the connection bandwidth available to a mobile device or the location of the user at the time.

4. Architecture of WISE Web Services

From the above, we can see that WISE web services can be built in several ways: placing agents on user devices, on web service servers, and on both sides of the connection. In this section, we describe each of these architectures.

4.1 Agents on WISE User Devices

The agents used in VBKM [18] exemplify the first WISE service architecture in which agents residing on mobile devices manage and utilize the context of the customers in order to assure effective services. This type of agents often embodies the characteristics of light processing (that might involve simple agent strategies) and not relating to other users of the designated web services.

Figure 1 shows a generic architecture for this type of WISE services. Without loss of generality, an agent at a mobile device is delegated to deal with a web service server (that encompasses *m* web services). This delegation involves management and utilization of the profiles, the context, and the strategies indirectly through a manager (that aims to coerce a coherent management and utilization).

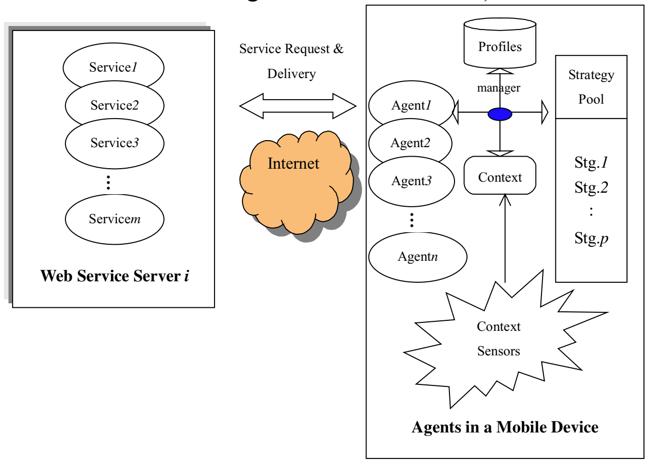

Figure 1. Architecture for WISE device agents

For each web service that is requested by a WISE device, the WSDL information of the service is kept at the mobile device. In addition, a WISE device also maintains certain QoS information about the service so that agents may use them to make more intelligent services. Information such as response time history (or simply the last response time), network bandwidth requirement (or usage) can be utilized to come up with various strategies when making service selections and delivery decisions.

4.2 Agents at a WISE Server

The agents deployed in the examples of CAVBMC [14] and MALCR [22] demonstrate the second WISE service architecture in which agents residing on the WS server capture and manage the context of the customers in order to assure more effective services. This type of agents often takes the forms of multiagent systems that embody the characteristics of heavy processing and might relate a user to other users of the designated web services.

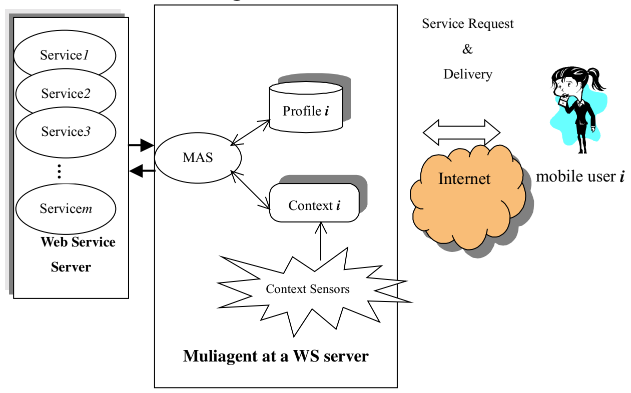

Figure 2. Architecture for WISE server agents

Figure 2 shows a generic architecture of WISE servers. The multiagent system (MAS) is delegated to deal with the web service server (that encompasses *m* web services). This delegation involves management and utilization of the profiles and the context of all of its customers. In addition, since the server needs to make sure all customers are happy, a QoS agent may be deployed to perform QoS service admission and management. The agent will inspect the QoS requirements of all customers and decide which customer should get how much resource to meet its QoS needs. On-line QoS monitoring is performed by each customer agent and reported back to the QoS agent so that it can make dynamic resource control.

The QoS capability of a WISE server may be published on its UDDI registry so that service requestors may decide if and which WISE server should be contacted for the desired service. Since QoS attributes are not part of the standard UDDI definition, only those clients that understand WISE QoS definition can talk to each other for QoS negotiation and delivery.

4.3 Agents on Both Ends

With WISE servers and WISE devices, web services may utilize yet another type of architecture, with software agents on both ends of web service connections. Such architecture opens door for very powerful multi-agent negotiation and cooperation scenarios. For example, it is possible for two or more agents to go into sophisticated bargain negotiations, involving rounds of offers and counter-offers, in order to search for the best deal. It is also possible to involve multiple user agents in auction-style negotiation, granting services to only those who are willing to pay a good price.

Rather than pursuing effort on multi-agent *negotiation*, we believe that multi-agent *cooperation* for QoS service and delivery provides a real opportunity for showing the real benefit of the WISE web service model. Since most users and applications have dynamic resource requirements at run-time, it is beneficial for servers to allocate only enough resources to applications at any time. Therefore dynamic QoS adjustment should be done aggressively to better utilize server resources and to achieve the best performances for all users. However, the WISE model is different from the traditional multi-agent model where all agents are assumed to be equally powerful. We envision that most WISE user devices have only very simple computing power. Therefore server agents should do most of the processing and make complex decisions and leave only rudimentary monitoring and responses to device agents.

5. Implementing WISE

One of the main goals for the WISE project is to provide working intelligent web services. Therefore, we must show that the WISE architecture can be efficiently implemented. In this section, we show a WISE web service implementation on top of J2EE web services and also the QoS web services protocol.

5.1 A WISE Implementation on J2EE

Web services are not tied to a specific platform like JVM or .NET since they focus on the protocols used to exchange messages, rather than the implementation that supports those protocols. However, since WISE is not part of the standard protocols, we must design our own WISE infrastructure support for each specific platform.

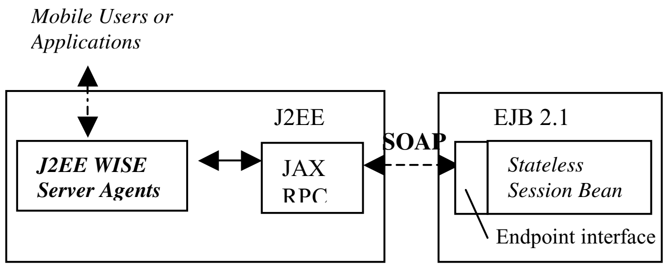

Figure 3. J2EE WISE server agents (non-WSDL)

J2EE provides a service-oriented infrastructure to automatically support and manage distributed components called Enterprise JavaBeans (EJBs). Enterprise developers can therefore concentrate on application components. J2EE web services (EJB 2.1) include web services APIs (e.g., JAX-RPC) that can be used to communicate with other web services and allow EJBs to act as web services. When developing a web service as a stateless session bean, an endpoint interface must be defined in order to generate the JAX-RPC client stubs (Figure 3), a WSDL document, or both. The JAX-RPC client stub can then be packaged with the WISE server agents in a J2EE client JAR and used to access the stateless session bean, using the SOAP protocol. If a WSDL document is generated from the endpoint interface, other SOAP toolkits such as the Microsoft SOAP Type Library (modeling either WISE server agents or device agents) can use the document to access the stateless bean (Figure 4).

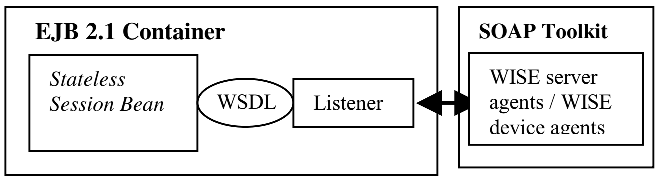

Figure 4. J2EE WISE sever/device agents (WSDL)

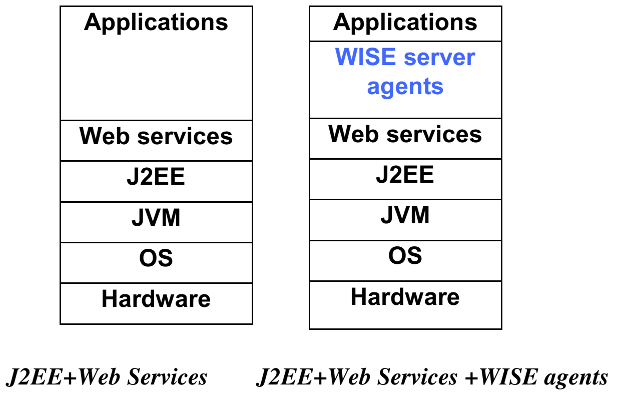

Figure 5. J2EE web service stacks vs WISE stacks

WISE web servers work on top of standard web service interface protocols, yet have very simple built-in intelligence and convenience. WISE server agents exert simple strategies to make their host web services perform better without a major effort or perception from users. Figure 3 shows a working WISE structure in which the WISE server agents provide lightweight intelligence (on behalf of mobile users or applications) to J2EE web services through the JAX-RPC client stubs. Figure 5 shows the WISE application infrastructure stack that includes the layer of WISE server agents to endow applications with "intelligent" web services (versus "plain" web services delivered by current application infrastructure stack). More details on our implementation and its performance will be reported in future reports.

5.2 WISE QoS Protocol

Many real-time web services have a short duration but need to have a reasonably firm guarantee on the performance. It is important to make sure that a minimum level of performance can be achieved. If the performance cannot be guaranteed, it is better to reject the request as early as possible so that the customer can try to locate another server that can perform the web service.

In WISE, we adopt a session-based QoS management [3] to reduce the overheads in making decisions. Every WISE device (or client) keeps a profile on the resource needs for each of the web services it may request. Every WISE server maintains its current QoS grant levels on all resources under its control. When a WISE device requests for a new service with QoS, the following steps are performed:

1. A WISE device sends to a WISE server the request for the establishment of a QoS web service with the desired range of QoS parameters.
2. The QoS agent on the WISE server performs the QoS admission process to check if the QoS request can be granted.
3. The server QoS agent responds with an offered QoS level to the WISE agent on the device notifying it the available resources at the time.
4. If the offered QoS level is acceptable, a web service *contract* is established. The WISE device then sends in the actual request to initiate the service.

The distinct features of the WISE QoS algorithm are the *efficiency* for making QoS admissions and the *adaptability* to the dynamic behaviors of WISE devices. The WISE QoS algorithm is based on the *share-driven* scheduling [19] that has been shown to be very effective for managing dynamic network traffic [5] and also multimedia computing [1]. Using the share-driven scheduling, each active session is committed a certain share (i.e. percentage) of the resource (CPU, network, *etc.*). This committed share is the minimum level of service that can be used by the session and will not be reduced regardless of the total load in the system. However, if a session has a certain unused share due to its lack of workload, this extra share will be used by all other sessions with active workloads waiting to be processed. The usage of the extra share among active sessions is divided in proportional to their previously-assigned shares. The will keep the system fully utilized at any time, avoiding any reserved but unused system time.

The WISE QoS algorithm is easy to implement and works very well for each resource managed independently. However, most web service applications may need QoS control for more than just one resource. For example, a multimedia session may require a sufficient capacity on both

CPU and memory. We are studying ways to integrate several resources in the WISE QoS management framework [19].

6. Conclusions

Web services have recently received much interest and support form industry to make the World Wide Web more service oriented. With the proposed standards such as SOAP, WSDL, UDDI and others, web services are becoming more ubiquitous. To make them even more "web-friendly", some lightweight intelligence can be inserted into the web service infrastructure.

In this paper we present the WISE web service architecture. WISE servers and devices are built on top of existing web service protocols but have simple web service intelligence to assist decision-makings. We have shown how WISE may be used to build intelligent web service delivery and QoS guarantee. The WISE web service architecture has been presented and QoS algorithm discussed. Our goal in this project is to show the benefit of deploying simple web service agents on the client (mobile) device and on servers. Rather than jumping into the *intelligent web* concept directly, which may still take many years to come, we believe the WISE approach will be much cheaper to implement and much easier to deploy.

7. References

[1] Abeni, L. and G. Buttazzo, Integrating Multimedia Applications in Hard Real-Time Systems, Proceedings of the IEEE Real-Time Systems Symposium, Madrid, Spain, pp. 4-13, December 1998.

[2] Cheng, John and Wellman, Michael, The walras algorithm: A convergent distributed implementation of general equilibrium outcomes. Computational Economics, Vol. 12(1), pp. 1-24, 1998.

[3] Cherkasova, L. and Phaal, P., Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites, IEEE Transactions on Computers, Vol. 51(6), pp. 669-685, 2002.

[4] Conitzer, V. and Sandholm, T., Vote Elicitation: Complexity and Strategy-Proofness, National Conference on Artificial Intelligence, 2002.

[5] Demers, A., Keshav, S., and Shenker, S., Analysis and Simulation of a Fair Queueing Algorithm. In Journal of Internetworking Research and Experience, pp.3-26. October 1990. Also in Proc. of ACM SIGCOMM'89, pp.3-12.

[6] Dey, A. K., Salber, D., & Abowd, G. D., A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16, 2001.

[7] Genesereth, M. R. and Ketchpel, S. P., Software Agents. Communications of the ACM, Vol. 37(7), pp. 48-53, 1994.

[8] Hendler, James., Agents and the Semantic Web, IEEE Intelligent Systems, Special Issue on the Semantic Web, Vol. 16, No. 2, pp. 30-37, March/April 2001.

[9] Janca, Peter. Pragmatic Application of Information Agents. BIS Strategic Decisions, Norwell, United States, 1995.

[10] Lee, Chen, John Lehoczky, Dan Siewiorek, Raj Rajkumar and Jeff Hansen, A Scalable Solution to the Multi-Resource QoS Problem. In *Proceedings of the 20th IEEE Real-Time Systems Symposium,* December 1999.

[11] Mani, Anbazhagan and Arun Nagarajan, Understanding quality of service for Web services, IBM Technical Report, Jan. 2002, "www6.software.ibm.com/software/developer/library/ws-quality.pdf"

[12] Menasce, Daniel A., QoS Issues in Web Services, IEEE Internet Computing, November-December, pp. 72-75, 2002.

[13] Parekh, A. K. and Gallager, R. G., A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case. IEEE/ACM Trans. Networking, Vol. 1, no.3, pp.344-357, June 1993.

[14] Peng, K. S. and Yuan, Soe-Tsyr, Location Based and Customerized Voice Information Service for Mobile Community, International Conference on Electronic Commerce, HongKong, 2002.

[15] Sandholm, T., Distributed Rational Decision Making, Multiagent Systems edited by Gerhard Weiss, The MIT Press, London, England, 1999.

[16] Sandholm, T, eMediator: A Next Generation Electronic Commerce Server, Computational Intelligence Vol. 18(4), pp. 656-676, 2002.

[17] Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., and Plaxton, G., A Proportional Share Resource Allocation Algorithm for Real-Time, Time-shared Systems. In Proc. of IEEE Real-Time Systems Symposium, pp.288-299, December 1996.

[18] Sun, J. W. and Yuan, Soe-Tsyr, Ontology-Based Task-Oriented Voice Mining for Mobile B2E Applications, *International Conference of Information Management Research and Practice*, Taiwan, 2002.

[19] Wang, Song and Kwei-Jay Lin, A General Resource Management Framework for Real-Time Operating Systems, Proc. International Conference on Parallel and Distributed Systems (ICPADS), Chungli, Taiwan, December 2002.

[20] Wang, Y.C. and Lin, K.J., Implementing a general real-time scheduling framework in the RED-Linux real-time kernel. In Proc. IEEE Real-Time Systems Symposium, Phoenix, Arizona, Dec 1999.

[21] Wellman, M. and Wurman, P., Market-aware agents for a multiagent world. Robotics and Autonomous Systems Journal, Vol. 24, pp.115-125, 1998.

[22] Yuan, Soe-Tsyr and Tsao, Y. W., A Recommendation Mechanism for Contextualized Mobile Advertising, *Expert Systems with Applications*, Vol. 24(4), pp. 399-414, 2003.