2003-109

Dup

TÜBİTAK

TÜRKİYE BİLİMSEL VE TEKNİK ARAŞTIRMA KURUMU

THE SCIENTIFIC AND TECHNICAL RESEARCH COUNCIL OF TURKEY

Elektrik, Elektronik ve Enformatik Araştırma Grubu

Electric, Electronics and Informatics Research Grant Committee

UDDI DEPOLARI İÇİN SANAL İŞ AKIŞI DESTEĞİ

PROJE NO : 101E041

PROF.DR. ASUMAN DOĞAÇ

MART 2003
ANKARA

Önsöz

UDDI (Universal Description Discovery and Integration) değişik şirketlerin internet üzerinden birbirlerini bulup birlikte çalışabilmelerini sağlamak için IBM ve Microsoft tarafından önerilmiş bir standard olup şirketlerin bilgileri UDDI deposunda tutulmaktadır. Bu depo şirketleri yaptıkları işe göre değişik kategorilere ayıran bir altyapıya sahiptir. Bu kategoriler sayesinde şirketler internet üzerinden otomatik olarak iş ortağı bulabilmektedir. Ayrıca şirketlerin sağladıkları servislerin nasıl kullanılacağını anlatan bir dokümanın internet adresi ve bu dokümanın kategorisi de UDDI deposunda tutulmaktadır. Böylece şirketler otomatik olarak iş ortaklarını bulup, onların servislerini kullanabilmektedir.

Bu projede elektronik ticaret konusunda yukarıda bahsedilen ve günümüz teknolojisi olan UDDI kullanımının yaygınlaştırılması ve kolaylaştırılması için bir otomasyon sistemi geliştirmek amaçlanmıştır.. Bu otomasyon sistemi için şirketlerin ürün ve fiyat bilgilerine ulaşabilmek için bir yöntem geliştirilmiş ve bu yöntem sayesinde kullanıcıdan alınan bilgilerle hangi şirketin hangi servislerinin kullanılacağının otomatik olarak bulunup bir iş akışı sistemi sayesinde uygun sıraya konup insan müdahalesi gerektirmeden çalıştırmak mümkün hale getirilmiştir.

Geliştirilen sistem şirketlerin UDDI deposuna kayıt yaptırarak internet üzerinden sağladıkları servisleri otomatik olarak bulması ve bu sayede değişik şirketlerin birbirinin yaptığı işlerden haberdar olmasını ve internet üzerinden iş ortağı bulmasını sağlamaktadır. Proje kapsamında bu sistem yazılımı sayesinde internet üzerinden herhangi bir şirket otomatik olarak kendine iş ortağı bulabilmektedir. Bu otomasyon literatüre getirilen en büyük yeniliktir. Bu projede yapılan çalışmalar sonucu elde edilen bilgilerin ve yazılımların kurumumuz bünyesinde devam eden ve önümüzdeki günlerde değerlendirmesi yapılmakta olan Avrupa projelerinde kullanılması amaçlanmaktadır. Geliştirilen yazılım bu projedelerdeki endüstriyel ortakların sistemlerine entegre edilerek uygulamaya aktarılması tasarlanmaktadır.

Bu proje Türkiye Bilimsel ve Teknik Araştırma Kurumu tarafından desteklenmiştir.

İÇİNDEKİLER

ÖZ

Projede elektronik ticaret konusunda günümüz teknolojisi olan UDDI (19, 27) kullanımının yaygınlaştırılması ve kolaylaştırılması için bir otomasyon sistemi geliştirmek amaçlanmıştır. Bu otomasyon sistemi için şirketlerin ürün ve fiyat bilgilerine ulaşabilmek için bir yöntem geliştirilmiş, ve bu yöntem sayesinde kullanıcıdan alınan bilgilerle hangi şirketin hangi servislerinin kullanılacağı otomatik olarak bulunup bir iş akışı sistemi sayesinde uygun sıraya konup insan müdahalesi gerektirmeden çalıştırılmıştır.

Geliştirilen sistem şirketlerin UDDI deposuna kayıt yaptırarak internet üzerinden sağladıkları servisleri otomatik olarak bulması ve bu sayede değişik şirketlerin birbirinin yaptığı işlerden haberdar olmasını ve internet üzerinden iş ortağı bulmasını sağlamaktadır. Proje kapsamında bu sistem yazılımı sayesinde internet üzerinden herhangi bir şirket otomatik olarak kendine iş ortağı bulabilmektedir. Bu otomasyon teknolojiye bir yenilik getirmiştir. Bu projede yapılan çalışmalar sonucu elde edilen bilgilerin ve yazılımların kurumumuz bünyesinde devam eden ve önümüzdeki şu sıralarda değerlendirmesi yapılmakta olan Avrupa projelerinde kullanılması amaçlanmaktadır. Geliştirilen yazılım bu projelerdeki endüstriyel ortakların sistemlerine entegre edilerek uygulamaya aktarılması tasarlanmaktadır.

Anahtar sözcükler: UDDI, İşakışı, WSDL, SOAP, Ağ Servisleri, DAML+OIL, RosettaNet, XML

ABSTRACT

This project has developed an automation system to increase and simplify the use of current UDDI (19, 27) registries. For this automation system, a method to access the product and price information of the companies has been developed. With the help of this method and the information taken from the user, it becomes possible to find the services of companies to be exploited using a workflow system. These services has beeen ordered and invoked without human intervention.

The developed system provides the facilities such as finding the web services of the companies, being aware of the businesses of other companies, by means of applying to the UDDI system. As a result of this project, by using the software developed any company can find a partner on the Internet automatically. This automation is a novelty for UDDI technology. We aim to use the software developed in this project in the European projects which are being carried out or evaluated currently. It is planned to pass the developed system into application by integrating it with the systems of industrial partners.

Keywords: UDDI, Workflow, WSDL, SOAP, Web Services, DAML+OIL, RosettaNet, XML

PROJE ANA METNİ

GİRİŞ

Proje UDDI depoları için şirketler arasında iş akışı otomasyonunu sağlamayı konu olarak ele almaktadır. Bu bağlamda bir iş akışı sistemi tasarlanıp ve geliştirilmiştir. Kullanıcıların istedikleri ürünleri verilecek olan kriterlere göre UDDI deposundan bulunup tüm işlemlerin otomatik olarak yapılabilmesi ve sonucun kullanıcıya iletilmesi için gerekli yazılımlar geliştirilmiştir.

GENEL BİLGİLER

UDDI (Universal Description Discovery and Integration) değişik şirketlerin internet üzerinden birbirlerini bulup birlikte çalışabilmelerini sağlamak için IBM ve Microsoft tarafından önerilmiş bir standard olup şirketlerin bilgileri UDDI deposunda tutulmaktadır (19, 27). UDDI şu an en çok kabul gören iki saklayıcıdan biridir. Bu alanda diğer önemli bir çalışma ebXML'dir (Electronic Business XML) (17). Hewlett Packard'da da benzer çalışmalar yürütülmektedir (18). Bu depo şirketleri yaptıkları işe göre değişik kategorilere ayıran bir altyapıya sahiptir. Bu kategoriler sayesinde şirketler internet üzerinden otomatik olarak iş ortağı bulabilmektedir. Ayrıca şirketlerin sağladıkları servislerin nasıl kullanılacağını anlatan bir dokümanın internet adresi ve bu dokümanın kategorisi de UDDI deposunda tutulmaktadır. Böylece şirketler otomatik olarak iş ortaklarını bulup, onların servislerini kullanabilmektedir. WSDL (Web Service Description Language) bu döküman tiplerinden en önemlisidir (29). WSDL yardımıyla şirketler sağladıkları servisleri kullandıkları programlama dilinden bağımsız olarak tanımlayabilmekte ve bu servisleri çalıştırmak için gerekli olan bilgiye ulaşılmasını sağlayabilmektedir. WSDL dökümanı SOAP (Simple Object Access Protocol) komutları içermektedir. SOAP internet üzerinden mesaj alış verişini ve uzaktan fonksiyon çağırmayı (remote procedure call) sağlamak için çıkarılmış bir standarttır (26). Şirketler bu standarda uygun olarak sağladıkları servisleri SOAP sunucusuna koyduklarında bu servis internet üzerinden çeşitli yazılımlar yardımıyla kullanılanılabilmektedir. Ancak, UDDI deposunda tutulan kategoriler yardımıyla aranan ürünün maliyetinin ne olduğu veya ürün kalitesi ve markası gibi satın almada önemli etkeni olan bilgilere ulaşılamamaktadır. Bu proje standarttaki bu eksikliği gidermek için bir otomasyon sistemi geliştirilmiştir.

GEREÇ

Proje kapsamında aşağıdaki gereçler kullanılmıştır:

1. IBM UDDI Saklayıcısı
2. IBM DB2 Veritabanı
3. WSTK Ağ Servisleri Gereçleri
4. Apache Tomcat Uygulama Sunucusu
5. Java Programlama Dili

YÖNTEM

Proje süresince yürütülen yönteme göre, ilk etapta geliştirilen yazılım parçalarının tasarımı yapılmıştır. Bu tasarım çerçevesinde sistemin temel özelliklerini taşıyan basit bir prototip yazılımı gerçekleştirilmiş, bu prototip irdelenerek tasarımdaki eksiklikler giderilerek tasarım geliştirilmiştir. Daha sonra bu tasarım çerçevesinde sistemin bütün özelliklerini içeren bir prototip yazılımı gerçekleştirilmiş, proje süresi boyunca literatür düzenli olarak takip edilip literatürdeki son gelişmeler çerçevesinde tasarımda gerekli düzenleme ve değişiklikler yapılıp bu değişiklikler geliştirilen prototiplere aktarılmıştır.

Bu yönteme ek olarak izlenmiş olan proje aşama ve zamanlama planı aşağıdaki gibidir:

Başlıca Aşamalar	**Ayrıntılı Bilgi**	**Zamanlama(*)**
UDDI deposuna örnek şirketlerin kaydedilmesi	Şirketler arasında örnek bir senaryo çıkarılacak ve bu senaryoya göre gerekli şirket ve servisler UDDI deposuna kaydedilecektir.	1-2
İş akışı sisteminin tasarımı ve yazılımı	İş akışı sisteminin tasarımı yapılacak ve iki aşamalı olarak yazılımı gerçekleştirilecektir	1-12
Kullanıcı profilini olusturan arayüzün geliştirilmesi	Kullanıcıdan alınması gereken bilgiler belirlenecek ve bu bilgilerden bir profil çıkaran bir arayüz geliştirilecektir.	2-12
Akıllı sunucunun tasarımı ve yazılımı	Kullanıcı profiline göre UDDI depolarından en uygun servisleri bulup bu servislerden bir iş akışı tanımı çıkaran bir sunucu tasarımı ve yazılımı.	5-12
Akıllı sunucu ile iş akışı sisteminin entegrasyonu	Akıllı sunucunun iş akışı sistemine entegre edilmesi için gerekli tasarım ve yazılım gerçekleştirilecektir.	9-12

BULGULAR

1) *UDDI deposuna örnek şirketlerin kaydedilmesi*: Örnek şirketlerin bulunmasının kolaylaştırılması için önce bir servis ontolojisi tanımlandı. Ortak olarak karar verilmiş bir ontoloji kullanımıyla web servisleri daha kolay bulunabilmektedir (20). Şu an literatürdeki en yaygın kullanılan ontoloji dili RDF (Resource Description Framework)(23,24) tabanlı DAML+OIL'dir (Darpa Agent Markup Language Ontology Interface Layer) (4, 5, 9). Bu projede servis ontolojisi olarak DAML+OIL tabanlı DAML-S kullanılmıştır (6, 7, 21). Aynı zamanda bu iş tabloda gösterildiği gibi projenin ilk iki ayını kapsamaktadır. Bu iş başarıyla tamamlanmıştır. Öngörülen senaryo şöyledir:
    - Senaryoda üç çeşit bilgisayar donanımı ilişkili şirket bir de ulaştırma şirketi bulunmaktadır:
        1. Bayi (Retailer)
        2. Distribütör
        3. Üretici
        4. Ulaştırıcı şirket

Bir müşteri toplu olarak bilgisayar parçası almaya karar verir bayie gider. Eğer bayide müşterinin istediği miktarda malzeme varsa isteği hemen bayi tarafından karşılanır. Eğer bu miktar bayide yoksa, bayinin bilgisayar sistemi UDDI deposundaki kendi belirleyeceği kıstaslardaki (fiyat, konum, vs.) tüm distribütörleri bulur ve onların kataloglarına bakarak bu miktarları karşılayıp karşılayamayacaklarını anlar. Daha sonra bayinin bilgisayar sistemi, bu miktarı karşılayabilcek olan distribütörlerin, distribütörlerin elinde olmaması ihtimaline karşın distribütörlerin üreticilerle ilişkiye girdiği ve bu malzemelerin yerlerine ulaştırma işlemlerini gerçekleştiren ulaştırıcı şirketinde bulunduğu dinamik olarak bir iş akışı çıkarır. Distribütörler aynı zamanda dağıttıkları malzemenin UNSPSC (Universal Standard Products and Services Classification) numarasına görede aratılabilir (28). Daha sonra bu iş akışı ilgili şirketlerin Internet üzerindeki servisleri iş akışındaki belirtildiği sıra ve dökuman alışverişi çerçevesinde gerçekleştirilerek malzemeler ilgili kişilerin eline geçer.

Senaryo geliştirildikten sonra bu senaryoyu gerçekleştirecek olan şirketlerin ve bu şirketlerin servislerinin özellikleri tasarlanmış ve UDDI deposuna, UDDI'nin verdiği API (Application Programming Interface, Uygulama Programlama Arayüzü) ile kaydedilmiştir. Projede tanımlanan şirketler RosettaNet'in tanımladığı iş akışı tanımlarını kullanmaktadır (25).

2) *İş akışı sisteminin tasarımı ve yazılımı:* Bu iş tabloda da belirtildiği gibi 12 ay yani tüm proje boyunca sürmekteydi. Proje döneminde proje başvurusunda da belirtilen iş akışı sisteminin tasarımı, ilk prototip ve genel sistem başarıyla geliştirilmiştir. İş akışında arada gidip gelen dökümanların formatını belirleyen diğer bir standart CBL dir (Common Business Library) (1). Bu projede önceden değinildiği gibi RosettaNet'in belirlediği format uygulanmıştır. Bu dökümanlar XML tabanlıdır (30). Bu formata uyulmasıyla geliştirilen sistem diğer aynı formata uygun sistemlerle konuşabilmesi sağlanmıştır. Bu formatların içinden gerekli bilgilerin çıkarılması için bir XML diline ihtiyaç duyulmaktadır. Şu an literatürde en çok kullanılan XML dilleri, XML-QL (8) ve Xquery'dir (31). Bu projede XML dili olarak XML-QL kullanılmıştır. Bu tasarımın en önemli elemanları iş akışını temsil eden iş akışı dili ve bu iş akışı dilini yorumlayıp, işleyebilen (sırası gelen servisi çalıştırıp gerekli dökuman transferini gerçekleştiren) programdır. İş akışı dili şu an Internet dökuman alışverişi standardı olarak benimsenen XML tabanlı bir dildir (2, 3). Kavramsal olarak bu dil bloklardan oluşmaktadır. Bu bloklar programlama dillerindeki yapılara benzemekte olup, bunlar IF, WHILE, ACTIVITY, ANDPARALLEL, SEND bloklarından oluşmaktadır. Bu blokların belli bir kareografide dizilimi ile istenilen iş akışı elde edilebilmektedir. Yukarıda anlatılan senaryodaki şirketlerin servisleri iş akışı bloklarından ACTIVITY bloklarıyla temsil edilmektedir. İş akışı sisteminin tasarımı kısaca aşağıdaki gibidir:

- Yukarıdaki senaryoda anlatıldığı gibi bayi bilgisayar sistemi dinamik olarak iş akışını belirtilen iş akışı dilinde oluşturduktan sonra bu iş akışını tasarlanan sisteme TCP/IP üzerinden göndermektedir. İş akışı sistemi, iş akışını aldıktan sonra, gerekli servislerin iş akışında belirtilen sırasıyla, WSDL tanımlarını UDDI deposundan alıp ve bu tanımı kullanarak bu servisleri çalıştıracak olan SOAP mesajlarını oluşturmaktadır (WSDL yardımıyla şirketler sağladıkları servisleri kullandıkları programlama dilinden bağımsız olarak tanımlayabilmekte ve bu

servisleri çalıştırmak için gerekli olan bilgiye ulaşılmasını sağlayabilmektedir. WSDL dökümanı SOAP (Simple Object Access Protocol) komutları içermektedir. SOAP internet üzerinden mesaj alış verişini ve uzaktan fonksiyon çağırmayı (remote procedure call) sağlamak için çıkarılmış bir standarttır. Şirketler bu standarda uygun olarak sağladıkları servisleri SOAP sunucusuna koyduklarında bu servis internet üzerinden çeşitli yazılımlar yardımıyla kullanılanılabilmektedir.) Daha sonra bu SOAP mesajları servislere gönderilerek ilgili servislerin çalıştırılması sağlanmıştır.

3) *Kullanıcı profilini olusturan arayüzün geliştirilmesi*: Örnek vermek gerekirse yukarıda anlatılan senaryoda, bayi sistemi distribütörleri ararken belirteceği kıstaslar bu profilin elemanlarını oluşturmaktadır. Profil sisteminin ana işlevlerini gerçekleştiren arayüz başarıyla geliştirilmiştir.
4) *Akıllı sunucunun tasarımı ve yazılımı*: Kullanıcı profiline göre UDDI depolarından en uygun servisleri bulup bu servislerden bir iş akışı tanımı çıkaran bir sunucu tasarımı ve yazılımı proje programına başarıyla gerçekleştirilmiştir.

TARTIŞMA/SONUÇ

Proje başarılı bir şekilde sonuçlandırılmıştır. Projede elde edilen bilgilerle (10, 11, 12, 13, 14, 15, 16, 22) yayınları yapılmıştır. Geliştirilmiş olan sistem şirketlerin UDDI deposuna kayıt yaptırarak internet üzerinden sağladıkları servisleri otomatik olarak bulması ve bu sayede değişik şirketlerin birbirinin yaptığı işlerden haberdar olmasını ve internet üzerinden iş ortağı bulmasını sağlamaktadır. Proje kapsamında bu sistem yazılımı sayesinde internet üzerinden herhangi bir şirket otomatik olarak kendine iş ortağı bulabilmektedir. Bu otomasyonun teknolojiye önemli bir katkısı olduğu görüşündeyiz. Bu projede yapılan çalışmalar sonucu elde edilen bilgilerin ve yazılımların kurumumuz bünyesinde devam eden ve önümüzdeki günlerde değerlendirmesi yapılacak olan Avrupa projelerinde kullanılmıştır. Geliştirilmiş olan yazılım bu projedeki endüstriyel ortakların sistemlerine entegre edilerek uygulamaya aktarılmıştır. Aynı zamanda bu projede edinen deneyim ile Avrupa Topluluğu 6. Çerçeve Programına "SebXML: Anlamsal olarak zenginleştirilmiş ebXML Saklayıcıları aracılığıyla Ağ Servislerinin kullanımı" adıyla bir kaynak proje sunulmuştur. Bu proje ile ilgili bilgi aşağıdaki tabloda gösterilmiştir.

Exploiting Web Services through Semantically Enriched ebXML Registries: SebXML

Lead Partner: Middle East Technical University, Software R&D Center, Ankara, Turkey

Abstract Web services have become a main thrust of the IT industry. There are two almost universally accepted standards for Web services: SOAP (Simple Object Access Protocol) for invoking services and WSDL (Web Services Description Language) for describing the technical specifications of the services. There are well-established service registries like UDDI (Universal Description, Discovery and Integration) by Microsoft and IBM, and ebXML (electronic business XML) by UN/CEFACT. The infrastructures for Web services are also readily available through well-established application servers like IBM's WebSphere, Microsoft's .NET Framework or BEA's Weblogic. All of these application servers provide support for SOAP, WSDL and UDDI connectivity.

What the IT industry needs today is semantic support for service discovery, composition and monitoring as well as semantic support for the privacy and security of the Web services. The currently available semantic support in service registries through taxonomies is inadequate since it is not possible to describe service properties through taxonomies. On the other hand, Web services like their real life counter parts may have several properties. Web services can only be exploited in their full potential when their properties are expressed in an interoperable and machine processable way. The solution is to use service ontologies to describe the semantics of Web services. DAML-S is a very important initiative in this respect defining an upper ontology. The research on exploiting service semantics is going on. However the performance of current initiatives for exploiting service semantics is not at an acceptable level to be adopted by the IT industry. Furthermore there are a number of issues that need to be addressed and research before Web services become the prominent paradigm for distributed computing such as improving performance of semantic service discovery of Web services and exploiting user context for service discovery. This project aims to address these issues. The objectives of the project are as follows:

- Improving the performance of exploiting service semantics by using the meta data and service discovery mechanisms of ebXML registries
- Enriching ebXML registries by developing efficient mechanisms to store and access OWL (Web Ontology Language) or DAML+OIL based service semantics in ebXML registries and providing facilities for service composition
- Developing mechanisms for exploiting user context for service discovery
- Developing a demonstrator for tourism industry

Keywords

Technology Service semantics, Service discovery, service composition, service orchestration, exploiting user context for service discovery, ebXML registries

Application Tourism

Partners

- Middle East Technical University (METU), Turkey
- IDI EIKON, Spain
- ALTEC, Greece
- INTRO Solutions, Turkey
- European Dynamics (ED), Greece
- Royal Melbourne Institute of Technology (RMIT), Australia

Duration	2 years
Type	Applied Research and Demonstration
Effort (PM)	120 (Project work)+12 (Administration)
Total Cost	Manpower: 120 PM= 600,000 Euro Travel: 50,000 Euro Hardware: 50,000 Euro Administration: 12 PM= 60,000 Euro Total: 760,000 Euro
Requested Cost	**470,000 Euro**
Liaison	A National Project (Turkey) supported by the Scientific and Technical Research Council of Turkey (TUBITAK): Exploiting Service Semantics through ebXML Registries A National Project (Turkey) supported by the Turkish Technology Development Foundation (TTGV): A Framework for Travel ecommerce Application Development A National Project Proposal (Australia) on a Web service and Agent-based infrastructure for tourism industry by Royal Melbourne Institute of Technology (RMIT)

Referanslar

1. Common Business Library (CBL) URL. http://www.xCBL.org/
2. F. Casati and M-C. Shan "Definition, Execution, Analysis and Optimization of Composite E-Services" IEEE Data Engineering Bulletin, Vol. 24, No.1, pp. 29-34.
3. F. Casati and M-C. Shan "Dynamic and Adaptive Composition of E-Services" Information Systems, to appear.
4. DARPA Agent Markup Language, http://www.xml.com/pub/a/2002/01/30/daml1.html
5. DAML+OIL, http://www.w3.org/2001/10/daml+oil
6. DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), DAML-S: Semantic Markup for Web Services, in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.
7. The DAML Services Coalition, "DAML-S: Web Service Description for the Semantic Web", to appear in The First International Semantic Web Conference (ISWC), Sardinia, Italia, June 9-12th, 2002.
8. Deutsch M. Fernandez D. Florescu A. Levy and D. Suciu "XML-QL: A query language for XML" W3C Document, http://www.w3.org/ TR/NOTE-xml-ql, 1998.
9. Denker, G., Hobbs, J. R., Narayan, S., Waldinger, R., "Accessing Information and Services on DAML-Enabled Web, Semantic Web Workshop, Hong Kong, China, 2001.
10. Dogac, A., Guest Editor, ACM Sigmod Record, Special Section on Data Management Issues in e-commerce, Vol. 31, No. 1, March 2002.
11. Dogac, A., Cingil, I., Laleci, G. B., Kabak, Y., 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002.
12. Dogac, A., Kabak, Y., Laleci, G., "A Semantic-Based Web Service Composition Facility for ebXML Registries", ICE2003 9th International Conference on Concurrent Enterprising, June 2003
13. Dogac, A., Laleci, G., Kabak, Y., Cingil, I.,"Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002, http://www.research.microsoft.com/research/db/debull/issues-list.htm.
14. Dogac, A., Tambag, Y., Pembecioglu, P., Pektas, S., Laleci, G. B., Kurt, G., Toprak, S., Kabak, Y., "An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs", ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 2002.
15. Dogac, A., Laleci, G., Kurt, G., Kabak, Y., Acar, A., "A Platform for Semantically Enriched Mobile Services" , in Proc. of the First International Conference on Mobile Business, Athens, Greece, July 2002.
16. Dogac, A., Tumer, A., "Issues in Mobile Electronic Commerce", Journal of Database Management, Vol. 13, No. 1, January 2002, pp. 37-43.
17. ebXML http://www.ebxml.org/
18. e-Speak http://www.e-speak.hp.com/
19. IBM UDDI registry http://www-3.ibm.com/services/uddi/find
20. McIlraith, S. A., Son, T. C., Zeng, H., "Semantic Web Services", IEEE Intelligent Systems, March/April 2001, pp. 46-53.
21. McIlraith, S. A., Son, T. C., Zeng, H., "Mobilizing the Semantic Web with DAML-Enaled Web Services", Semantic Web Workshop 2001, Hongkong, China.
22. Ozen, B., Kilic, O., Altinel, M., Dogac, A. "Highly Personalized Information Delivery to Mobile Clients", KLUWER, Wireless Networks, the Journal of Mobile Communication, Computation and Information.

23. Resource Description Framework (RDF) Schema Specification 1.0 W3C Candidate Recommendation, http://www.w3.org/TR/CR-rdf-schema, 2000.
24. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax, 1999.
25. RosettaNet http://www. rosettanet.org.
26. Simple Object Access Protocol (SOAP) http://www.w3.org/TR/SOAP/
27. Universal Description, Discovery and Integration (UDDI) www.uddi.org
28. Universal Standard Products and Services Classification (UNSPSC) http://eccma.org/unspsc
29. Web Service Description Language (WSDL) http://www.w3.org/TR/wsdl
30. XML, "Extensible Markup Language (XML) 1.0", W3C Recommendation, http:// www.w3.org/ TR/REC-xml-19980210, 1998.
31. XQuery, "An XML Query Language 1.0", W3C Working Draft, http://www.w3.org/TR/2002/WD-xquery-20020430.

Exploiting Web Service Semantics: Taxonomies vs. Ontologies

Asuman Dogac, Gokce Laleci, Yildiray Kabak, Ibrahim Cingil

Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

Abstract

Comprehensive semantic descriptions of Web services are essential to exploit them in their full potential, that is, discovering them dynamically, and enabling automated service negotiation, composition and monitoring. The semantic mechanisms currently available in service registries which are based on taxonomies fail to provide the means to achieve this. Although the terms "taxonomy" and "ontology" are sometimes used interchangably there is a critical difference. A taxonomy indicates only class/subclass relationship whereas an ontology describes a domain completely. The essential mechanisms that ontology languages provide include their formal specification (which allows them to be queried) and their ability to define properties of classes. Through properties very accurate descriptions of services can be defined and services can be related to other services or resources.

In this paper, we discuss the advantages of describing service semantics through ontology languages and describe how to relate the semantics defined with the services advertised in service registries like UDDI and ebXML.

1 Introduction

When looking towards the future of web-services, it is predicted that the breakthrough will come when the software agents start using web-services rather than the users who need to browse to discover the services. Currently well accepted standards like Web Services Description Language [WSDL] and Simple Object Access Protocol [SOAP] make it possible only to "dynamically access" to Web services in an application. That is, when the service to be used is known, its WSDL description can be accessed by a program which uses the information in the WSDL description like the interface, binding and operations to dynamically access the service. However to dynamically *discover* services, say through software agents require detailed semantic information about the services to be available.

Currently, a number of taxonomies are being used to discover services in service registries like [UDDI] or [ebXML]. The most widely used taxonomies are North American Industrial Classification Scheme [NAICS] for associating services with "industry" semantics; Universal Standard Products and Services Classification [UNSPSC] for classifying product/services and [ISO 3166] for locale.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

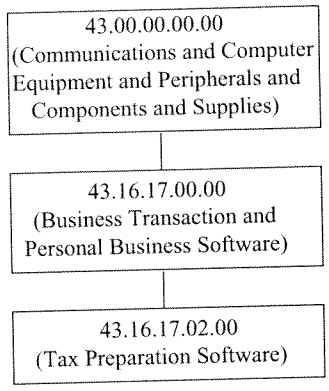


Figure 1: A part of an UNSPSC Taxonomy

A taxonomy is a hierarchy and a unique code is usually assigned to each node of the hierarchy. This code also encodes its path. For example, as shown in Figure 1, UNSPSC code for *Communications and Computer Equipment and Peripherals and Components and Supplies* is "43.00.00.00.00". One of the classes under this class is *Business transaction and personal business software* whose code is "43.16.17.00.00". Going one more level deep,"43.16.17.02.00" is the code for *Tax preparation software*.

By relating a service with the codes in such taxonomies, it is possible to give the service a certain amount of semantics. For example, when a service uses the code "43.16.17.02.00" to describe its semantics, we understand that the service is about "tax preparation software". Therefore, a user looking for a service related with a "tax preparation software" can search the service registries with the corresponding UNSPSC code to obtain all services that have declared themselves to be related with this UNSPSC code. Note however that these services could be anything; they may be "selling" such software; they may be providing it as a service. Furthermore the service may have any number of properties like certain service qualities, say, minimum and maximum service delivery times; may require certain payment obligations, say, advance credit card payment. Additionally a service may be available at a discount price when it is used in aggregation with the other services that the company provides. There can be several such properties of a service and hence the user has to go through the services found to manually pick the service that satisfies her requirements. Notice for example that, in UDDI registries, this information can only be made available informally (i.e., with no formal semantics) through "OverviewDocs" or "OverviewURLs" of the tModels associated with the services. In short, taxonomies do not help in this respect.

It follows that to exploit the Web services to their full potential we need more powerful tools, that is, ontologies to describe their semantics. In fact, currently, describing the semantic of Web in general [Berners-Lee], and semantic of Web services in particular are very active research areas. There are a number of efforts for describing the semantics of Web services such as [McIlraith a, McIlraith b, Denker]. Among these [DAML-S] defines an upper ontology, that is, a generic "Service" class. In order to make use of DAML-S upper ontology, the lower levels of the ontology need to be defined. To provide interoperability, application domains must share such specifications. In fact, an ontology describes *consensual knowledge*, that is, it describes meaning which has been accepted by a group not by a single individual. Standard bodies need to define domain specific ontologies.

In this paper, we discuss the advantages of defining service semantics through ontology languages. We further note that, once the semantic is defined, it is also necessary to relate the defined semantics with the services advertised in the service registry. Therefore we describe how this can be achieved in UDDI and ebXML registries.

The paper is organized as follows: Section 2 briefly introduces DAML-S upper ontology and describes the

advantages of ontology languages over taxonomies. Section 3 describes mechanisms provided by UDDI and ebXML registries for associating semantics with the services advertised.

2 Service Semantics through Ontology Languages

Web services, like their real life counterparts, may have many properties. The aim of this section is to demonstrate that all the necessary properties of services can easily be defined through an ontology language. While developing domain specific ontologies, it is a good idea to ground them in upper ontologies since in this way they are more consistent and it becomes easier to integrate them within distributed heterogeneous systems.

DAML-S provides such an upper ontology, that is, it defines a top level "Service" class with some generic properties common to most of the services. The "Service" class has the following three properties:

- *presents:* The range of this property is *ServiceProfile* class. That is, the class Service *presents* a *ServiceProfile* to specify what the service provides for its users as well as what the service requires from its users.
- *describedBy:* The range of this property is *ServiceModel* class. That is, the class Service is *describedBy* a *ServiceModel* to specify how it works.
- *supports:* The range of this property is *ServiceGrounding*. That is, the class Service *supports* a *ServiceGrounding* to specify how it is used.

DAML-S is based on [DAML+OIL] and DAML+OIL allows very sophisticated ontologies to be defined and queried through, for example, DAML APIs. The queries on ontologies usually access the properties of classes or traverse the ontology. Hence it is possible to standardize queries to facilitate their use in an automated way.

In the following we provide examples for some of the properties of DAML-S Service Profile:

- *serviceParameters:* In DAML-S, service parameters denote an expandable list of RDF properties that may accompany a profile description. The range of each property is unconstrained, i.e. no range restrictions are placed on the service parameters as shown in the following:

```
<daml:Property rdf:ID="serviceParameter">
 <daml:domain rdf:resource="&service;#ServiceProfile"/>
 <daml:range rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
</daml:Property>
```

- *degreeOfQuality:* This property of Service Profiles provide qualifications about the service.

```
<daml:Property rdf:ID="degreeOfQuality">
 <daml:domain rdf:resource="&service;#ServiceProfile"/>
 <daml:range rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
</daml:Property>
```

2.1 An Example Service Ontology

In this section we define an example ontology grounded in DAML-S for tax services for the sole purpose of describing the power of ontology languages. A detailed description of several properties of services including the methods of charging and payment, the channels by which the service is requested and provided, constraints on temporal and spatial availaility, service quality, security, trust and rights attached to a service, is given in [O'Sullivan]. Through the example ontology we show how some of these properties can be defined.

As shown in Figure 2, the top level class of this ontology is "TaxServices" which inherits from DAML-S "Service" class. In this way, several properties of the "TaxServices" class are conveniently defined by inheriting from the properties of the DAML-S "Service" class. For example, "paymentMethod" is defined as a subproperty

```
<!DOCTYPE uridef[
 <!ENTITY rdf      "http://www.w3.org/1999/02/22-rdf-syntax-ns">
 <!ENTITY rdfs     "http://www.w3.org/2000/01/rdf-schema">
 <!ENTITY xsd      "http://www.w3.org/2000/10/XMLSchema">
 <!ENTITY daml     "http://www.daml.org/2001/03/daml+oil">
 <!ENTITY profile  "http://www.daml.org/services/daml-s/2001/05/Profile.daml">
 <!ENTITY service  "http://www.daml.org/services/daml-s/2001/05/Service.daml">
 <!ENTITY tp       "http://www.srdc.metu.edu.tr/2002/10/TaxPayment.daml">
 <!ENTITY unspsc   "http://www.eccma.org/unspsc/browse/43.html">

<rdf:RDF
      xmlns:rdf =      "&rdf;#"
      xmlns:rdfs =     "&rdfs;#"
      xmlns:xsd =      "&xsd;#"
      xmlns:daml =     "&daml;#"
      xmlns:profile =  "&profile;#"
      xmlns:service =  "&service;#"
      xmlns:unspsc =   "&unspsc;#"
      xmlns:tp =       "&tp;#"

 <daml:Ontology rdf:about=" ">
   <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
   <daml:imports rdf:resource="&service;">  </daml:Ontology>

 <daml:Class rdf:ID="TaxServices">
   <rdfs:subClassOf rdf:resource="&service;"/> </daml:Class>

 <rdf:Property rdf:ID="paymentMethod">
    <rdfs:subPropertyOf rdf:resource=&profile;serviceParameter/>
    <rdfs:domain rdf:resource="&service;#ServiceProfile"/>
    <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>

 <rdf:Property rdf:ID="serviceGuarantee">
    <rdfs:subPropertyOf rdf:resource=&profile;degreeOfQuality/>
    <rdfs:domain rdf:resource="&service;#ServiceProfile"/>
    <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>

 <rdf:Property rdf:ID="requiredService">
   <rdfs:domain rdf:resource="&damls;#Service/>
   <rdfs:range rdf:resource="&damls;#Service/> </rdf:Property>

 <rdf:Property rdf:ID="discountAmount">
   <rdfs:domain rdf:resource="#PromotionRequirements"/>
   <rdfs:range rdf:resource="&xsd;#nonNegativeInteger"/> </rdf:Property>

 <daml:Class rdf:ID="TaxPreparationService">
   <rdfs:subClassOf rdf:resource="#TaxServices"/>
   <rdfs:label> Tax Preparation Service </rdfs:label>
   <rdfs:subClassOf>
     <daml:Restriction>
       <daml:onProperty rdf:resource="#promotion"/>
       <daml:toClass rdf:resource="#PromotionRequirements"/>
     </daml:Restriction> </rdfs:subClassOf> </daml:Class>

 <rdf:Property rdf:ID="promotion">
    <rdfs:subPropertyOf rdf:resource=&profile;serviceParameter/>
    <rdfs:domain rdf:resorce="#TaxPreparationService"/>
    <rdfs:range rdf:resource="#PromotionRequirements"/> </rdf:Property>

 <daml:Class rdf:ID="PromotionRequirements">
    <rdfs:subClassOf rdf:resource="#TaxServices"/>
    <rdfs:subClassOf>
     <daml:Restriction>
       <daml:onProperty rdf:resource="#requiredService"/>
        <daml:toClass rdf:resource="&damls;#Service"/>
     </daml:Restriction> </rdfs:subClassOf> </daml:Class>

 <daml:Class rdf:ID="LegalConsulting">
   <rdfs:subClassOf rdf:resource="#TaxServices"/>
   <rdfs:label> Legal Tax Consultancy Service </rdfs:label>
 </rdfs:subClassOf> </daml:Class>

 <rdf:Property rdf:ID="serviceCodeUNSPSC">
   <rdfs:label> Defining a property to denote UNSPSC codes of services</rdfs:label>
   <rdfs:domain rdf:resource="&damls;#Service/>
   <rdfs:range rdf:resource="&unspsc;#UNSPSCCodes"/> </rdf:Property>
 </rdf:RDF>
```

Figure 2: An Example Ontology Description for Tax Payment Domain

of DAML-S "ServiceProfile serviceParameter" property and "serviceGuarantee" as a subproperty of DAML-S "ServiceProfile degreeOfQuality" property. Notice that, in this generic ontology, ranges of these properties are defined to be the most general class in a DAML+OIL ontology, that is, *daml+oil#Thing* class. This class is specialized when defining ontology instances as shown in Figure 3. This general ontology also states that a

```
<!DOCTYPE uridef[
 <!ENTITY rdf      "http://www.w3.org/1999/02/22-rdf-syntax-ns">
 <!ENTITY tp       "http://www.srdc.metu.edu.tr/2002/10/TaxPayment.daml">

<rdf:RDF
     xmlns:rdf =      "&rdf;#"
     xmlns:tp =       "&tp;#"

<tp:TaxPreparationService rdf:ID="TaxHeavenTaxPreparationService">
  <tp:serviceCodeUNSPSC>43.16.17.02.00 </tp:serviceCodeUNSPSC>
  <tp:paymentMethod> CreditCard </tp:paymentMethod>
  <tp:serviceQuarantee>24</tp:serviceQuarantee>
  <tp:promotion rdf:resource="#MyPromotionRequirements"/>
</tp:TaxPreparationService>

<tp:PromotionRequirements rdf:ID="MyPromotionRequirements">
  <tp:requiredService rdf:resource="#LegalConfort"/>
  <tp:discountAmount>10 </tp:discountAmount>
</tp:PromotionRequirements>

<tp:LegalConsulting rdf:ID="LegalConfort">
</tp:LegalConsulting>
</rdf:RDF>
```

Figure 3: Service Ontology of the company "TaxHeaven"

"promotion" property can be associated with "TaxPrepatationService" which requires another service to be used and specifies the discount amount when the two services are used in aggregation. Finally, the ontology states that the UNSPSC codes may be associated with services.

Figure 3 shows a specific instance of the generic ontology given in Figure 2, for the company "TaxHeaven". "TaxHeaven" is providing a tax preparation service. The service has the maximum delivery time of 24 hours; and accepts only credit card payment. "TaxHeaven" also provides a seperate legal consultancy service for tax payment, called "LegalConfort" and provides 50% discount for this service to the users of their tax preparation service, namely, "TaxHeavenTaxPreparationService".

To provide ease in readability, these ontologies are also shown graphically in Figure 4. The graphical representation followed is inspired by [Gonzalez-Castillo].

3 Associating Semantics with Service Registries

In this section we describe how to associate the semantic defined with services advertised by using the mechanisms provided by UDDI and ebXML registries.

3.1 UDDI Registries

The mechanism to relate semantics with services advertised in the UDDI registries are the tModel and the catagory bags of registry entries. tModels provide the ability to describe compliance with a specification, a concept, a shared design or a taxonomy. Services have category bags and any number of tModel keys can be put in these category bags.

In relating the semantics defined in DAML+OIL with the services advertised in UDDI registries, the first question to be answered is where to store the semantic descriptions. Generic descriptions can be stored by the standard bodies who define them and the server, where the service is defined, can host the semantic description of the service instance. This facilitates the maintenance of the service descriptions. However there are times, when it is necessary to query all the individual service descriptions. Therefore a combined schema per industry domain containing all the semantic descriptions of the services pertaining to this domain may be necessary to facilitate global querying.

The second issue is relating the ontology defined in DAML+OIL with the services advertised in the UDDI registry. For this purpose, similar to WSDL, DAML+OIL should be classified as "DAMLSpec" with "uddi-org:types" taxonomy. A seperate tModel of type "DAMLSpec" for the combined schema of each industry

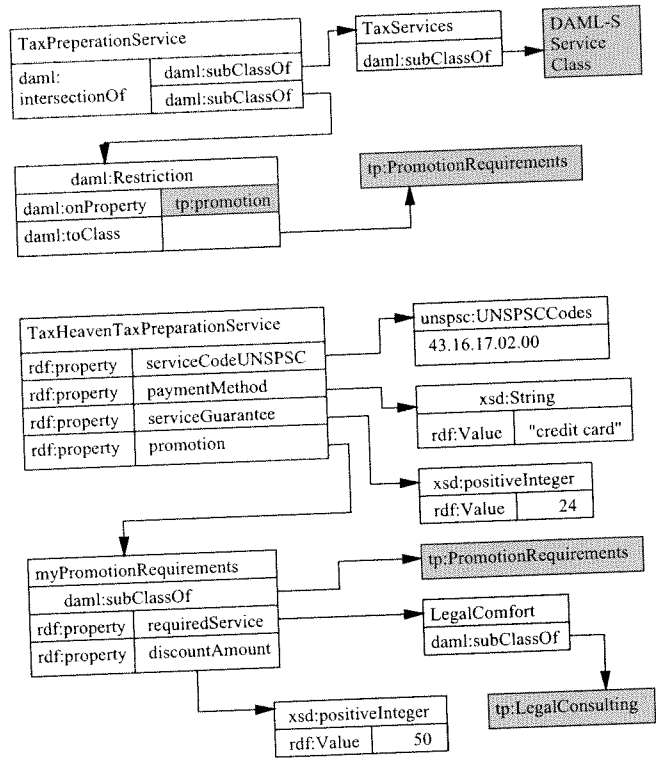


Figure 4: A pictorial description of "TaxHeaven" ontology

domain should be created. The services in an industry domain must contain the key of this tModel in their category bags. Hence, this tModel key can be used to find services in an industry domain through the UDDI registry.

There may be times where it is necessary to find all the instances of a generic class. For example, to choose a tax payment service with required functionality, it may be necessary to retrieve all semantic tax payment service descriptions to check their properties. To be able to do this, that is, in order to find instances of a generic class, it is necessary to associate a tModel key for each generic service class and store this tModel key with individual service descriptions.

Finally, there should be a tModel key for each service instance. This tModel key can be used in searching the UDDI registry to find a particular advertised service instance according to its semantic description. A more detailed treatment of this issue is given in [Dogac].

3.2 ebXML Registries

The basic mechanism in ebXML registries for associating semantics with the objects stored in the registry is the "classification" hierarchy, called *ClassificationScheme*. *ClassificationScheme* defines a hierarchy of *ClassificationNodes*. The nodes in this hierarchy are related with registry objects through *Classification* objects. A *Classification* instance classifies a RegistryObject instance by referencing a node defined within a particular classification scheme. As an example, assume that "TaxHeavenTaxPreparationService" stored in the ebXML registry and a *ClassificationScheme* exits for the "Tax Payment" industry like the one provided in Figure 2. Figure 5 demostrates how this service is associated with this classification schema by using the classification object.

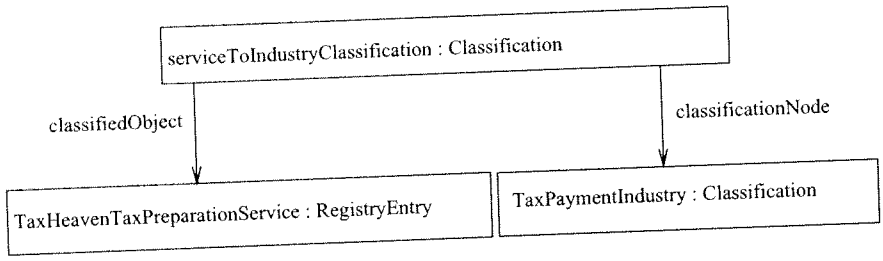


Figure 5: Associating a Service with a Classification Node in ebXML

Note that a registry object can be classified according to any number of classification schemes.

However classification structure provided by ebXML is not adequate to store DAML+OIL ontologies and need to be extended to be used for this purpose. Also ebXML registry interface needs to be extended to query DAML+OIL ontologies.

References

[Berners-Lee] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.

[DAML+OIL] DAML+OIL, http://www.w3.org/2001/10/daml+oil

[DAML-S] DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), DAML-S: Semantic Markup for Web Services, in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.

[Denker] Denker, G., Hobbs, J. R., Narayan, S., Waldinger, R., "Accessing Information and Services on DAML-Enabled Web, Semantic Web Workshop, Hong Kong, China, 2001.

[Dogac] Dogac, A., Cingil, I., Laleci, G. B., Kabak, Y., "Improving the Functionality of UDDI Registries through Web Service Semantics", 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002.

[ebXML] ebXML, http://www.ebxml.org/

[Gonzalez-Castillo] Gonzalez-Castillo, J., Trastour, D., Bartolini, C., "Description Logics for Matchmaking of Services", Technical Report, HP Labs, Bristol, UK.

[ISO 3166] ISO 3166, http://www.iso.ch/iso/en/prods-services/iso3166ma/index.html

[McIlraith a] McIlraith, S. A., Son, T. C., Zeng, H., "Semantic Web Services", IEEE Intelligent Systems, March/April 2001, pp. 46-53.

[McIlraith b] McIlraith, S. A., Son, T. C., Zeng, H., "Mobilizing the Semantic Web with DAML-Enaled Web Services", Semantic Web Workshop 2001, Hongkong, China.

[NAICS] North American Industrial Classification Scheme (NAICS) codes http://www.naics. com.

[SOAP] Simple Object Access Protocol (SOAP) http://www.w3.org/TR/SOAP/

[O'Sullivan] O'Sullivan, J., Edmond, D., Hofstede, A., "What's in a Service? Towards Accurate Description of Non-Functional Service Properties", in the Journal of Distributed and Parallel Databases, Vol. 12, No. 2/3, Sept./Nov. 2002.

[UDDI] Universal Description, Discovery and Integration (UDDI), http://www.uddi.org/.

[UNSPSC] Universal Standard Products and Services Classification (UNSPSC) http://eccma.org/unspsc

[WSDL] Web Service Description Language (WSDL), http://www.w3.org/TR/wsdl

Improving the Functionality of UDDI Registries through Web Service Semantics

Asuman Dogac, Ibrahim Cingil, Gokce Laleci, and Yildiray Kabak

Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
asuman@srdc.metu.edu.tr

In this paper we describe a framework for exploiting the semantics of Web services through UDDI registries. As a part of this framework, we extend the DAML-S upper ontology to describe the functionality we find essential for e-businesses. This functionality includes relating the services with electronic catalogs, describing the complementary services and finding services according to the properties of products or services. Once the semantics is defined, there is a need for a mechanism in the service registry to relate it with the service advertised. The ontology model developed is general enough to be used with any service registry. However when it comes to relating the semantics with services advertised, the capabilities provided by the registry effects how this is achieved. We demonstrate how to integrate the described service semantics to UDDI registries.

1 Introduction

Web services are modular and self-describing applications that can be mixed and matched with other Web services to create business processes and value chains. Recently, there have been a number of initiatives related with service discovery and composition lead by IT companies and consortiums like e-Speak [11] from HP, UDDI [19] from IBM and Microsoft, and ebXML [10] from United Nations/CEFACT and OASIS. Furthermore, HP has developed a platform, called eFlow [3, 4], for specifying, enacting, and monitoring composite Web services.

However the lack of standard business semantics creates inefficiencies in exploiting the Web service registries. Describing the semantics of Web services provides the ability for automatic Web service discovery, invocation, composition and interoperation, and Web service execution monitoring [7]. Recently there is an important iniative in this respect, namely, DAML-S [7]. DAML-S is a comprehensive effort based on DAML+OIL [5, 6] defining an upper ontology for Web services.

In this paper, we describe a framework for Web service semantics where we exploit the upper ontology defined by DAML-S to extend it with the functionality we find essential for the e-businesses and integrate it with UDDI registries.

Universal Description, Discovery and Integration (UDDI) is jointly proposed by IBM, Microsoft and Ariba. It is a service registry architecture that presents a

standard way for businesses to build a registry, discover each other, and describe how to interact over the Internet. Conceptually, the information provided in UDDI registries consist of three components: "white pages" of company contact information; "yellow pages" that categorize businesses by standard taxonomies; and "green pages" that document the technical information about services.

Currently there are no mechanisms to describe the metadata of services in UDDI. For example, locating parties that can provide a specific product or service at a given price, which we believe is an essential functionality of the e-businesses, is currently not available in UDDI. We give the following example to motivate the reader for the work presented in this paper:

Assume that a business user in Ankara wishes to buy second hand IBM desktop computers for the cheapest price that she can get, for over a period of time (that is, she wishes to establish a long term business relationship). The user also wishes to find services for possible products that may add value to the desktop, for example, a scanner. There is a need for the purchases to be delivered, and therefore complementary services like "delivery" are also necessary.

This business user can find a standard products and services code (like UNSPSC [20]) for desktops and the geography code for Ankara, and search for businesses in a UDDI registry. However there are a number of problems in this process:

- First, the user has to go through all the businesses found to check their services. These services could be anything related with desktops, not only the services that sell desktops. Therefore the user has to go through all the services found to distinguish the ones that "sell" desktops. With the projected near-term population of several hundred thousand to million distinct entities in a UDDI registry, it is unlikely that even this result set will be manageable.
- Second, it is not possible in UDDI to enforce a relationship between the service names and their functionality. Note that Web service description languages like WSDL only provide the signature of the operations of the service, that is, the name, parameters and the types of parameters of the service. Trying to discover services by name may not be always very meaningful since a service name could be anything and in any language. So it is not easy to figure out which of the services in the UDDI registry indeed realize the "sell" functionality.
- Third, among the services discovered that sell desktops there is no hint on which of these services actually sell "IBM" desktops and also their prices, since UDDI does not provide a mechanism to discover services on the basis of product instances. In other words, although it is possible to find the services according to the category of the products, it is not possible to find services by giving specific product information like their brand names or prices.
- Notice that the user is looking for a service that has a property: the service should deal with "second hand" products. There is no way to find such services since it is not possible to define properties for the services in UDDI.
- Products may have attributes that cannot be defined in product taxonomies like UNSPSC. For example given an anchor product, say a desktop, there

could be a number of products that add value to this product, say a scanner or a printer, and the user may wish to find the services related with these products as in the case of our example.
- Since there is no mechanism to define relationships among service types, it is hard to identify complementary services. Continuing with our example, the user cannot locate a complementary "delivery" service for the reason stated.

These limitations are not inherent in the UDDI specification, but rather stem basically from the lack of semantic descriptions of the Web services. Currently, describing the semantic of Web in general [1], and semantic of Web services in particular are very active research areas. There are a number of efforts for describing the semantics of Web services such as [13, 14, 9]. Among these DAML-S is a comprehensive effort defining an upper ontology.

In defining an ontology for e-businesses, there are certain functionality that we believe should be present. The first one is describing a standard way of relating the Web services with electronic catalogs. Also, we believe it is necessary to discover services with complementary functionality. For example a "delivery" service may be complementary to a "sell" service, and it should be possible to discover the related complementary "delivery" service instances given a "sell" service instance.

There is another issue to be handled once the semantics is defined: There should be a mechanism in the service registry to relate the semantics defined with service advertised. We also address this issue.

The paper is organized as follows: In Section 2, we very briefly summarize the ontology language used in this paper, namely, DAML-S. Section 3 describes the service ontology we propose. In this section, we also show how to integrate the proposed framework to UDDI. Section 4 concludes the paper.

2 DAML-S: Semantic Markup for Web Services

DAML-S [7], which is based on DAML+OIL [6], defines an upper ontology for describing service semantics and the top level class is the *Service* class. *Service* class has three properties:

- *presents* The class Service *presents* a *ServiceProfile* to specify what the service provides for its users as well as what the service requires from its users; that is, it gives the type of information needed by a service-seeking agent to determine whether the service meets its needs. *ServiceProfile* class has properties to describe the necessary inputs and outputs used or generated by a service; preconditions and postconditions, and binding patterns. Other properties of the *ServiceProfile* include *serviceParameter* to define parameters of services like maximum response time; *serviceType* to refer to a high level classification of services, such as B2B or B2C; and *serviceCategory* to refer to an ontology of services.
- *describedBy* The class Service is *describedBy* a *ServiceModel* to specify how it works.

– *supports* The class Service *supports* a *ServiceGrounding* to specify how it is used. A service grounding specifies the details of how an agent can access a service. It should be noted that DAML-S ServiceGrounding specification overlaps with WSDL [8].

3 The Proposed Semantic Framework

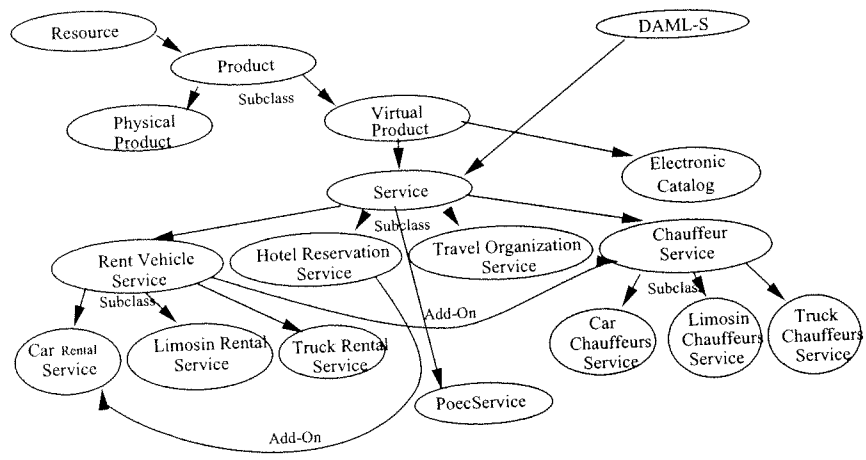


Fig. 1. An example service taxonomy

In order to better expolit the service registries, service semantics need to be defined. In developing ontologies for e-business applications, there are some important functionality that must be provided by Web services in a standard way, such as:

– *Discovering services with complementary functionality:* For example, a "Sell" service can complement a "Delivery" service. A standard property, say, "addOn" can be used to describe such complementary services to help with their discovery.
– *Finding services according to the properties of services or products:* Given an anchor product, a customer may be intersted in products that add value to this product. As an example, a customer willing to buy a computer, may also be willing to buy a "scanner" or a "ups", and may therefore be interested in finding services providing these products.
Also services themseves may have properties; for example a "sell" service may be dealing with only "second hand" products; or a "payment" service may have properties such as "Credit Card Payment".

– *Relating the services with electronic catalogs:* For example a user may not only wish to rent a car but a specific model like "Chevrolet Model 1956". Unless the instances of a "Car_Rental_Service" provide a standard way to access electronic catalogs, providing the user what he wants in an automated way, may become impossible.

We extend DAML-S upper ontology to provide this functionality. We further note that most of the services are related with products, let it be physical products or virtual products, i.e., information. Specifying services and products in the same taxonomy has the following advantages: the properties applicable to both physical and virtual products, such as "addOn" property is defined only once at the top level product class and inherited by the subclasses. Secondly, in this way it becomes possible to relate services and products in the same ontology, in a compact way. We use DAML+OIL as the ontology language both because of its richer semantic modelling primitives and to be able to be compatible with DAML-S.

In the proposed framework, both the services and the products are classified in a single top level taxonomy as shown in Figure 1. That is, under the top level class defined to be "resource" by RDF, we define a "Product" class that has "Physical Products" and "Virtual Products" as subclasses. The service class we define, namely "PoecService", is a subclass of "Virtual Products". In the following subsections we describe how the required functionality is achieved through the proposed semantic framework.

3.1 Discovering the services according to their functionality

We will first describe how a service ontology is used to find the services with desired functionality. A "generic" type service in an ontology defines service functionality; that is, an ontology associates generic classes with well-defined meanings. In order to discover the services according to their functionality, the generic service class of the desired functionality should be known. In other words, the generic service class name is the input for discovering services according to their functionality. Once the generic class is known, it is necessary to query the ontology to obtain all the subclasses of the given generic class as well as the "implementation" instances. The implementation instances thus found satisfy the required functionality.

To differentiate between the "generic" type services in the ontology from the service instances, we use DAML-S "serviceType". "PoecService" inherits both from the "Service" specification of DAML-S and from the "Virtual_Product" specification. "PoecService" class restricts the DAML-S "serviceType", and defines the type of the service to be a "generic" type or an "implementation" instance as shown in the following:

```
<daml:Class ID="PoecService">
  <rdfs:subClassOf rdf:resource="#Virtual_Product"/>
  <rdfs:subClassOf rdf:resource="http://www.daml.org/services/daml-s/2001/05/Service.daml"/>
  <rdfs:subClassOf>
    <daml:Restriction>
```

```
      <daml:onProperty rdf:resource="&profile;#serviceType"/>
      <daml:toClass rdf:resource="#ServiceTypes"/>
    </daml:Restriction> </rdfs:subClassOf> </daml:Class>
<daml:Class ID="ServiceTypes">
  <daml:oneOf rdf:parseType="daml:collection">
    <ServiceTypes rdf:ID="Generic"/>
    <ServiceTypes rdf:ID="Implementation"/>
  </daml:oneOf> </daml:Class>
```

An "implementation" instance gives the description of a particular service implementation. Consider the example service taxonomy provided in Figure 1. The services presented in this schema are generic services such as "Car_Rental_Service". Any implementation of this service, such as "My_Car_Rental_Service" declares itself to be an instance of the "Car_Rental_Service" class. Then, in order to find the implementations of "Rent_Vehicle_Service", it is necessary to query the ontology to find all subclasses of this class and all of their implementations. Since the queries involved have well defined goals like finding all subclasses of a given class, they can easily be standardized. It should be noted that DAML+OIL documents can be queried through DAML APIs or through any XML query language since the serialization syntax of DAML+OIL documents are XML. However, to the best of our knowledge, there are no DAML+OIL query languages yet.

3.2 Relating services with complementary functionality

Consider the ontology given in Figure 1. Assume that a customer needs a driver after renting a car. Given the generic name of the car rental service in the ontology, "Car_Rental_Service", to find out what kind of add on services are available for this service, we need to find out the add on services of this class and the add on services of all of its super classes. Such a search (through a query mechanism) will give us the "Chauffeur" service as the complementary (add on generic service) to the "Rent_Vehicle_Service" service. Now in order to find drivers for the car we plan to rent, we need to search the UDDI registry for services that contain the tModel for "Chauffeur" in their catagory bags. This issue is further elaborated in Section 3.5.

The "Added_Value" and "AddOn_To" properties defined in the following are used to express complementary services:

```
<rdf:Property ID="AddOn_To">
    <rdfs:domain resource="#Product"/>
    <rdfs:range  resource="#Product"/> </rdf:Property>
<rdf:Property ID="Added_Value">
    <rdfs:domain resource="#Product"/>
    <rdfs:range  resource="#Product"/> </rdf:Property>
```

Notice that since "PoecService" is a subclass of "Product", the "Added_Value" and "AddOn_To" properties are applicable to both services and products. Further note that, add on products of a super class are also add on products for all of its subclasses due to the inheritance in class hierarchies.

As noted previously, the process of finding the properties of services can be automated through a set of standardized queries to traverse DAML+OIL

descriptions for the super classes of a given class and retrieving the necessary properties.

3.3 Discovering the Services according to the attributes of Product Types

The attributes of the products can be used to discover the services in a similar way as described in Section 3.2. For example, a user, after locating a service for an anchor product, may want to discover the services providing add on products. The "Add-On" property defined in Figure 1 is used to express complementary products. Notice that these properties are applicable to both services and products.

Given an anchor product, in order to find the services for add on products, it is necessary to find the super classes of this anchor product, since add on products of its super classes are also add on products of this class. For example, the super class of "desktop" is "computer" which declares "scanner" as its add on product. This process can be automated through standardized queries as mentioned previously. Once the add on products are discovered, the services for these products can be obtained from the UDDI registries by using their UNSPSC codes.

3.4 Relating Services with Product Instances

Discovering services related with a specific product is of strategic importance for Web services. For example, a user may wish not only to rent a car but a specific model like "Chevrolet Model 1956". In such a case it is necessary to find car rental services that rent this specific model. However in order to discover services according to product instance information, it is necessary to form a relationship between the two.

To be able to associate service implementations with their related product instances, we first define a class called "ElectronicCatalog" to be a subclass of "Virtual_Product". Electronic catalog has the following properties, defined as a subproperty of "input" property of DAML-S ServiceProfile class:

- Catalog Schema Type
- Catalog Schema
- Catalog URI

These are, we believe, the minimum set of properties for an electronic catalog to be queried automatically. The range of these properties are the most generic class DAML Thing, which can be restricted in the individual schemas.

We then define a "QueryCatalog" standard service as a subclass of "PoecService". The services that provide a "QueryCatalog" service declare this through "has_Query_Catalog" service which is a sub property of "serviceParameters" property of DAML-S ServiceProfile class. The "QueryCatalog" class has the following properties: "inputCatalog", "inputQuery" (defined as subproperties of DAML-S "ServiceProfile input" property) and "QueryResult" (subproperty of DAML-S "ServiceProfile output") as shown in the following:

```
<daml:Class ID="ElectronicCatalog">
    <rdfs:subClassOf resource="#Virtual_Product"/> </daml:Class>
<rdf:Property rdf:ID="CatalogURI">
   <rdfs:subPropertyOf rdf:resource="&profile;#input"/>
   <rdfs:domain rdf:resource="#ElectronicCatalog"/>
   <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>
<rdf:Property rdf:ID="CatalogSchema">
   <rdfs:subPropertyOf rdf:resource="&profile;#input"/>
   <rdfs:domain rdf:resource="#ElectronicCatalog"/>
   <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>
<rdf:Property rdf:ID="CatalogSchemaType">
   <rdfs:subPropertyOf rdf:resource="&profile;#input"/>
   <rdfs:domain rdf:resource="#ElectronicCatalog"/>
   <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>
<daml:Class ID="QueryCatalog">
   <rdfs:subClassOf rdf:resource="#PoecService"/>
   <rdfs:subClassOf>
       <daml:Restriction>
         <daml:onProperty rdf:resource="&profile;#inputCatalog"/>
         <daml:toClass rdf:resource="#ElectronicCatalog"/>
       </daml:Restriction> </rdfs:subClassOf> </daml:Class>
<rdf:Property ID="has_Query_Catalog">
    <rdfs:subPropertyOf rdf:resource="serviceParameters"/>
    <rdfs:domain rdf:resource="&service;#ServiceProfile"/>
    <rdfs:range rdf:resource="&poec;QueryCatalog"/> </rdf:Property>
<rdf:Property rdf:ID="inputCatalog">
   <rdfs:subPropertyOf rdf:resource="&profile;#input"/>
   <rdfs:domain rdf:resource="#QueryCatalog"/> </rdf:Property>
<rdf:Property rdf:ID="inputQuery">
   <rdfs:subPropertyOf rdf:resource="&profile;#input"/>
   <rdfs:domain rdf:resource="#QueryCatalog"/>
   <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>
<rdf:Property rdf:ID="QueryResult">
   <rdfs:subPropertyOf rdf:resource="&profile;#output"/>
   <rdfs:domain rdf:resource="#QueryCatalog"/>
   <rdfs:range rdf:resource="&daml;#Thing"/> </rdf:Property>
```

To provide interoperability, the queries and the electronic catalogs must conform to standards. Possible standards for electronic catalogs include the Common Business Library (CBL) [2] catalog definition or RosettaNet Technical Dictionary [17]. Possible query standards to be used depends on the catalog structure. For Common Business Library and RosettaNet Technical Dictionaries, since they are defined in XML [22], XQuery [23] is a possible candidate. Note that for catalogs with a well-known schema like RosettaNet Technical Dictionary, there is no need to specify the Catalog Schema. However if the catalog is on a database, it is necessary to provide the database schema.

When it comes to service invocation, DAML-S specifies service grounding to describe how the service is used. In this respect, DAML-S specification is overlaping with Web Services Description Language (WSDL). WSDL is a well established standard for describing the interface and binding information of Web services. DAML-S service grounding specification has the same functionality as WSDL except for preconditions and post conditions for executing Web services. Yet this information is also available from DAML-S service profile definitions.

3.5 Relating Service Ontology with Service Instances

It is also necessary to provide the DAML+OIL descriptions of the service instances. The server where the service is defined can host the DAML+OIL de-

scription of service implementation instance. Storing a DAML+OIL description individually in this way isolates the description of each implementation instance and facilitates their maintenance by the service providers. However there are times, when it is necessary to query all the individual service descriptions, and this implies accessing all of them one by one, which may be inefficient. Therefore a combined schema per industry domain containing all the descriptions of the services pertaining to this domain may be necessary to facilitate querying. Note that the combined schema needs to be updated to contain the newly registered services.

There is one more issue to be handled to exploit the semantics defined for the Web services: there should be a mechanism in the service registry to relate the semantics defined with the service advertised. The semantic framework proposed can be integrated with UDDI as follows: Similar to WSDL, DAML+OIL Schema should also be classified as "damlSpec" with "uddi-org:types" taxonomy. A seperate tModel of type "damlSpec" can be created for the combined schema of each industry domain, and OverviewDoc element of the corresponding tModel can be made to point at the combined schema. The services in an industry domain contain the key of this tModel in their category bags and the OverviewDoc elements associated with these tModel keys point at the DAML+OIL description of service instances.

Furthermore, a tModel should be assigned to each generic service as well as service implementations. We therefore define the following in the common schema:

```
<daml:UniqueProperty rdf:ID="tModelKey">
  <rdfs:domain rdf:resource="#PoecSercive"/>
  <rdfs:range rdf:resource="&xsd;#decimal"/>
</daml:UniqueProperty>
```

4 CONCLUSIONS

When looking towards the future of web-services, it is predicted that the breakthrough will come when the software agents start using web-services rather than the users who need to browse, discover and compose the services. Among the challenges this breakthrough involves is the semantics of Web services.

Although some progress has been made in the area of web service description and discovery, and there are some important standards like SOAP, WSDL, and UDDI, and efforts for defining semantics like DAML-S; there is still more work needed in this area.

In this paper, we extend the DAML-S upper ontology to describe some functionality that we find essential for e-businesses like discovering services with complementary functionality, discovering services according to the properties of products or services and relating services with electronic catalogs. We then describe how to use the defined semantics through UDDI registries.

Our future work includes extending this work to ebXML [10] registries. ebXML registries allow to store classification hierarchies and relate registry items with classification nodes through classification objects. This feature of ebXML

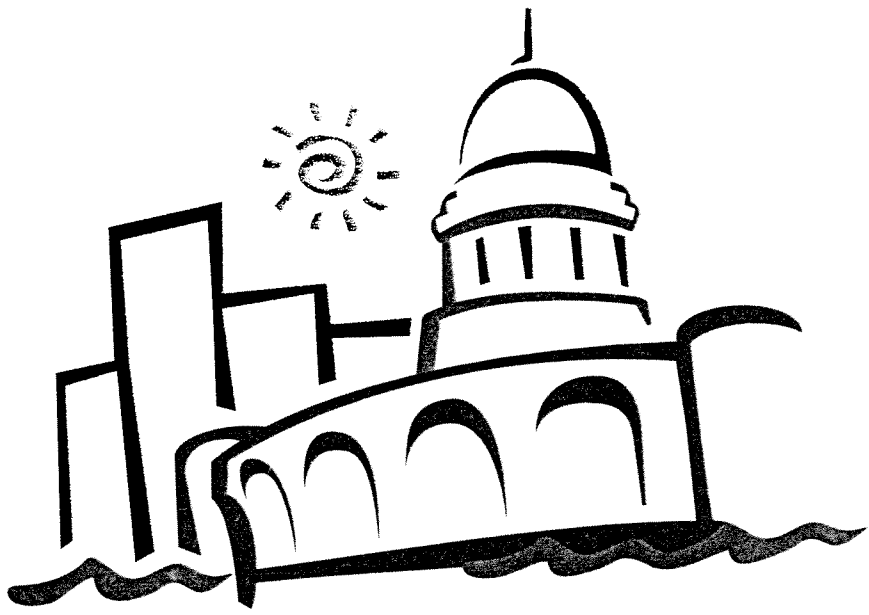
facilitates associating semantics with the services. However classification structure provided by ebXML is not adequate to store DAML+OIL ontologies and need to be extended to be used for this purpose. Also ebXML registry interface needs to be extended to query DAML+OIL ontologies.

References

1. Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.
2. Common Business Library (CBL) URL. http://www.xCBL.org/
3. F. Casati and M-C. Shan "Definition, Execution, Analysis and Optimization of Composite E-Services" IEEE Data Engineering Bulletin, Vol. 24, No.1, pp. 29-34.
4. F. Casati and M-C. Shan "Dynamic and Adaptive Composition of E-Services" Information Systems, to appear.
5. DARPA Agent Markup Language, http://www.xml.com/pub/a/2002/01/30/daml1.html
6. DAML+OIL, http://www.w3.org/2001/10/daml+oil
7. DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), DAML-S: Semantic Markup for Web Services, in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.
8. The DAML Services Coalition, "DAML-S: Web Service Description for the Semantic Web", to appear in The First International Semantic Web Conference (ISWC), Sardinia, Italia, June 9-12th, 2002.
9. Denker, G., Hobbs, J. R., Narayan, S., Waldinger, R., "Accessing Information and Services on DAML-Enabled Web, Semantic Web Workshop, Hong Kong, China, 2001.
10. ebXML http://www.ebxml.org/
11. e-Speak http://www.e-speak.hp.com/
12. IBM UDDI registry http://www-3.ibm.com/services/uddi/find
13. McIlraith, S. A., Son, T. C., Zeng, H., "Semantic Web Services", IEEE Intelligent Systems, March/April 2001, pp. 46-53.
14. McIlraith, S. A., Son, T. C., Zeng, H., "Mobilizing the Semantic Web with DAML-Enaled Web Services", Semantic Web Workshop 2001, Hongkong, China.
15. Resource Description Framework (RDF) Schema Specification 1.0 W3C Candidate Recommendation, http://www.w3.org/TR/CR-rdf-schema, 2000.
16. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax, 1999.
17. RosettaNet http://www. rosettanet.org.
18. Simple Object Access Protocol (SOAP) http://www.w3.org/TR/SOAP/
19. Universal Description, Discovery and Integration (UDDI) www.uddi.org
20. Universal Standard Products and Services Classification (UNSPSC) http://eccma.org/unspsc
21. Web Service Description Language (WSDL) http://www.w3.org/TR/wsdl
22. XML, "Extensible Markup Language (XML) 1.0", W3C Recommendation, http://www.w3.org/ TR/REC-xml-19980210, 1998.
23. XQuery, "An XML Query Language 1.0", W3C Working Draft, http://www.w3.org/TR/2002/WD-xquery-20020430.

SIGMOD 2002

Proceedings of the ACM SIGMOD International Conference on Management of Data
June 3-6, 2002



Madison

Edited by Michael Franklin
Bongki Moon and Anastassia Ailamaki

ACM SIGMOD

An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs*

Asuman Dogac, Yusuf Tambag, Pinar Pembecioglu, Sait Pektas,
Gokce Laleci, Gokhan Kurt, Serkan Toprak, Yildiray Kabak

Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

ABSTRACT

Today's Internet based businesses need a level of interoperability which will allow trading partners to seamlessly and dynamically come together and do business without ad hoc and proprietary integrations. Such a level of interoperability involves being able to find potential business partners, discovering their services and business processes, and conducting business "on the fly". This process of dynamic interoperation is only possible through standard B2B frameworks. Indeed a number of B2B electronic commerce standard frameworks have emerged recently. Although most of these standards are overlapping and competing, each with its own strenghts and weeknesses, a closer investigation reveals that they can be used in a manner to complement one another.

In this paper we describe such an implementation where an ebXML infrastructure is developed by exploiting the Universal Description, Discovery and Integration (UDDI) registries and RosettaNet Partner Interface Processes (PIPs). ebXML is an ambitious effort and produced detailed specifications of an infrastructure both for B2B and B2C e-commerce. However a public ebXML compliant registry/repository mechanism is not available yet. On the other hand, UDDI's approach to developing a registry has been a lot simpler and public registries are available. In ebXML, trading parties collaborate by agreeing on the same business process with complementary roles. Therefore there is a need for standardized business processes. In this respect, exploiting the already developed expertise through RosettaNet PIPs becomes indispensable. We show how to create and use ebXML "Binary Collaborations" based on RosettaNet PIPs and provide a GUI tool to allow users to graphically build their ebXML business processes by combining RosettaNet PIPs. In ebXML, trading parties reveal essential information about themselves through Collaboration Protocol Profiles (CPPs). To conduct business, an agreement between parties is necessary and this is expressed through a Collaboration Protocol Agreement (CPA). To help with this process, a tool is implemented to automate the process of configuring a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs). Being ebXML compliant mandates the messages exchanged to conform to ebXML messaging architecture. Furthermore since ebXML business processes may include several "Binary Collaborations", there is a need for a workflow enactment service to keep track of the execution. As a part of the infrastructure developed, we provide a server implementation which handles ebXML messages as well as keeping track of the data and control flow in the ebXML business processes.

*This work is supported in part by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 101E041

ACM SIGMOD '2002 June 4-6, Madison, Wisconsin, USA

1. INTRODUCTION

As businesses move more and more to the Web, it becomes increasingly important to improve the mechanisms of doing business over the Web. This basically involves developing standard ways of discovering the potential partners and automating the business processes among them.

Electronic business processes include both interactions between trading partners (public processes) and the private processes within a company (private processes). A public business process involves predefined message and document formats exchanged between the partners, the sequence of message exchange as well as some other mechanisms like the transport and security protocols.

A number of B2B standards have emerged [11] supporting the discovery and automation of public business processes over the Web. The basic functionalities of some of these standards are as follows:

- RosettaNet [15] provides a framework and predefined business processes called Partner Interface Processes (PIPs) for IT supply chain partners. This is a successful standard that has become a model for other industries as well. Compliance with RosettaNet framework implies automation of business processes among the partners over the Web. Although RosettaNet provides a Business Dictionary, since this is not a public registry, it is difficult to discover other potential partners. In fact the emphasis of RosettaNet framework is not transacting business "on-the-fly" but rather automating business among existing partners.
- UDDI [16], on the other hand, provides a public registry to discover businesses and their services. However

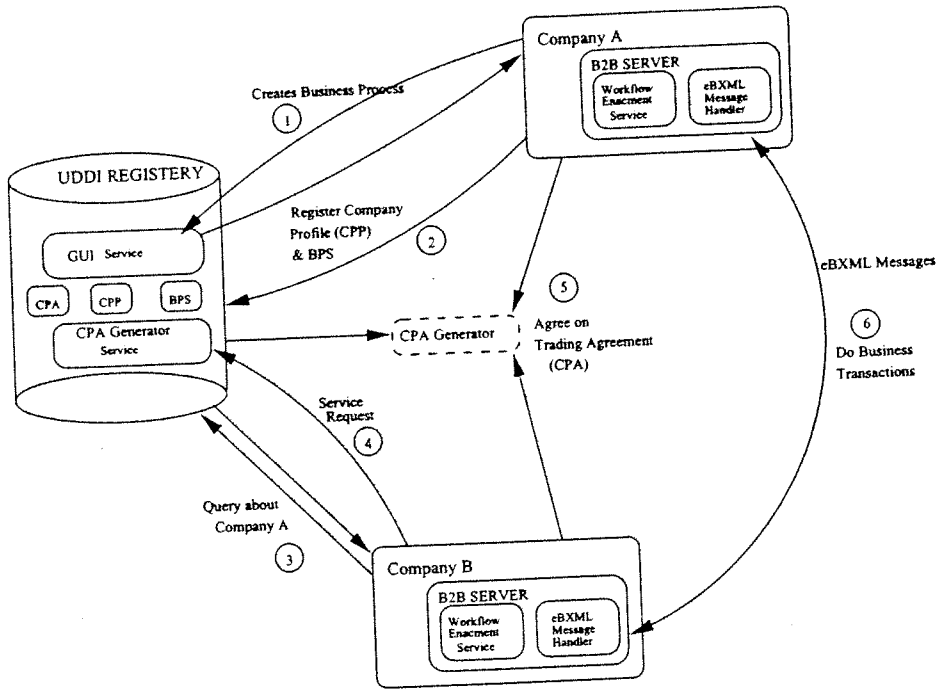


Figure 1: An Overview of the System Architecture

UDDI concentrates on the services and does not address public business processes explicitly. A service is an atomic unit of processing; whereas a business process involves collaboration and hence the exchange of a sequence of messages according to a predefined choreography of interactions with well defined roles for the involved partners.

- Whereby ebXML [4] allows arbitrary public business processes to be defined in XML and complements the framework with a registry where both trading partners and their public business processes can be discovered. ebXML also has a comprehensive message passing architecture that relies on SOAP with attachments and provides for the security and reliability of the messages. Furthermore ebXML defines a repository and well defined interfaces to query it. Unfortunately a public ebXML registry/repository is not available yet.

These B2B interoperation frameworks are competing and at the same time overlapping standards each with its own strengths and weaknesses. In this paper, we show how these standards can be used to complement one another by using the powerful aspects of each. The infrastructure we have developed exploits RosettaNet's well defined public processes; UDDI's public registry and ebXML's secure, reliable message passing mechanism. The infrastructure also provides a number of add-on services to the combined framework to facilitate conducting business on the Web.

The over all architecture and the contributions of the system can be summarized as follows:

- *The well defined RosettaNet PIPs are used to describe ebXML public business processes.* In ebXML, trading parties collaborate by agreeing on the same business process with complementary roles. If every partner develops its own business process, collaboration becomes very difficult. Therefore there is a need for standardized business processes. In this respect, exploiting the already developed expertise by RosettaNet through PIPs becomes indispensable.

  In order to use RosettaNet PIPs as ebXML business processes, it is necessary to map the PIP definitions into ebXML business process documents conforming to the DTD provided by ebXML. Since RosettaNet PIPs are defined in Unified Modelling Language (UML), it is possible to automate this conversion. In fact there are tools, such as [12] that generate XML output from UML diagrams through XML Metadata Interchange [21]. On the otherhand mapping RosettaNet PIPs into ebXML processes manually is not a difficult task. We show how to map ebXML public business processes into the RosettaNet Partner Interface Processes (PIPs).

- *A GUI tool is developed which allows users to graphically build their ebXML business processes by combining RosettaNet PIPs.* A RosettaNet PIP corresponds to a "Binary Collaboration" in ebXML. Business processes, however, can include more than one "Binary Collaboration" and there could be more than two party involved (multiparty collaboration). Therefore there is a need for a tool to help with the design of the choreography of multiple "Binary Collaborations" into a business scenario. We provide a GUI which allows users to graphically built their ebXML business processes by combining RosettaNet PIPs. The tool is registered to UDDI as a service to facilitate its discovery.

- *A UDDI registry is used to store ebXML documents like business processes, CPPs and CPAs and mechanisms are provided to facilitate their discovery.* In

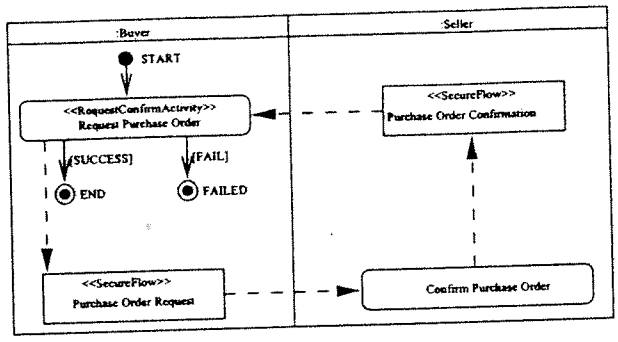


Figure 2: RosettaNet Business Process Flow Diagram for PIP3A4

UDDI, businesses and services can specify the categories to which they belong in their category bags to facilitate their discovery. An item in a category bag contains a tModel key. tModels provide the ability to describe compliance with a specification, a concept, or a shared design. When a particular specification is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification.

There are more than a hundred PIPs and a trading party may include any number of these PIPs into its business process specification (Note however that the order of PIPs can not be arbitrary; for example it does not make sense if a Purchase Order Request (PIP3A4) follows an Invoice Notification (PIP3C3)).

To facilitate the discovery of a business process, we have defined "tModels" for each of the RosettaNet PIPs. A business process specification includes the tModels of the involved PIPs into its catagory bag. In this way a trading party can discover other parties with compliant business processes by checking their catagory bags to see whether they contain the same PIPs as its own.

- *A tool is provided to automate the process of configuring a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs).* In ebXML a CPA is derived from the intersection of two CPP's. A Collaboration Protocol Profile (CPP) in ebXML framework contains essential information about the trading partner like contact information, industry classification, supported business processes and messaging service requirements. A CPA describes the *Messaging Service* and the *Business Process* requirements that are agreed upon by both partners. Therefore to configure a CPA, the tool matches *Process Specification, Roles, Transport and Transport Security* and *Document Security* levels of the CPPs. This tool is stored as a service in the UDDI registry.

- *A B2B server is developed to provide mechanisms for properly processing ebXML messages as well as keeping track of the data and control flow in the ebXML business processes.* ebXML business partners need to have software components at their sites to automatically process incoming ebXML messages as well as keeping track of business scenarios they are involved in. We developed a generic B2B Server implementation for this purpose. The server provides an ebXML Messaging Service infrastructure and workflow capabilities to keep track of business processes. The tool itself is available as a service from the UDDI registry.

Figure 1 demonstrates the overall architecture of the system. To be able to use the infrastructure developed, the trading parties need to install the B2B server implementation at their sites. This server has two main functionalities: handling the ebXML messages which basically involves preparing, sending, receiving and routing the messages, and keeping track of data and control flow in ebXML business processes. The next step to become an ebXML compliant business is to create ebXML business processes. We provide a GUI tool to help with this step which is available as a service from the UDDI registry. The graphical tool helps combine ebXML "Binary Collaborations" to define complex, multiparty ebXML business processes. These "Binary Collaborations" are obtained from RosettaNet PIPs. The trading parties can use this tool to form their business process. To become ebXML compliant, a business also needs to create its CPP and point it to the business process defined. Once all the necessary documents, that is, CPPs, business processes and business documents are created, they are registered to the UDDI registry.

The business parties can use UDDI registry to discover the potential ebXML compliant parties and their CPPs. To facilitate the discovery process we associate a tModel with each PIP and store the related tModel keys in the category bags of business processes.

Once the potential partners are discovered, it is necessary to reach an agreement on the way to carry out the business transactions. It is the Collaboration Protocol Agreement (CPA) that specifies the details of how two trading parties have agreed to conduct business electronically. A CPA is formed by combining the involved CPPs. We also provide a tool to facilitate this step which tries to match corresponding elements like Process Specification, Transport and Roles in CPPs to create a CPA.

After each company configures their B2B servers, the system becomes ready for operation and the parties can conduct their business transactions in an ebXML compliant way.

The paper is organized as follows: Section 2 gives an overview of the B2B standards addressed in this work, namely, RosettaNet, UDDI and ebXML. In Section 3 we

Message Exchange Controls - Request Purchase Order								
#	Name	Time to Acknowledge Receipt Signal	Time to Acknowledge Acceptance Signal	Time to Respond to Action	Included in Time to Perform	Is Authorization Required?	Is Non-Repudiation Required?	Is Secure Transport Required?
1.	Purchase Order Request Action	2 hrs	N/A	24 hrs	Y	Y	Y	Y
1.1.	Receipt Acknowledgment	N/A	N/A	N/A	Y	Y	Y	Y
1.2.	Purchase Order Confirmation Action	2 hrs	N/A	N/A	Y	Y	Y	Y
1.2.1.	Receipt Acknowledgment	N/A	N/A	N/A	N	Y	Y	Y

Figure 3: RosettaNet Message Exchange Controls for PIP3A4 (Request Purchase Order)

show how to map RosettaNet PIPs into ebXML business processes. Section 4 presents the GUI Tool developed for flexibly constructing ebXML business processes from RosettaNet PIPs. Registering ebXML documents to UDDI registries is described in Section 5. Section 6 summarizes the tool automating the process of configuring the CPA given two CPPs. The details of the B2B server is presented in Section 7. Finally Section 8 concludes the paper.

2. AN OVERVIEW OF THE B2B FRAMEWORKS ADDRESSED

A B2B interoperability standard, in general, involves the description of the message formats exchanged (e.g. purchase order), bindings to transport protocols (e.g. HTTP), the sequencing (e.g. after sending a purchase order message, an acknowledgment message must be received), the process (e.g. after a purchase order is accepted, the goods must be delivered to the buyer), and the security to be provided (like encryption, non-repudiation) [1]. Currently there are many Business-to-Business (B2B) electronic commerce standards based on XML [11]. The aim of this section is to briefly summarize the B2B electronic commerce frameworks addressed in this paper, namely, RosettaNet, UDDI and ebXML.

2.1 RosettaNet

Founded in 1998, RosettaNet [15] is an independent, self-funded, non-profit consortium dedicated to the development of XML-based standard electronic commerce interfaces to align the processes between supply chain partners on a global basis. The RosettaNet consortium includes IT companies like IBM, Microsoft, EDS, Netscape, Oracle, SAP, Cisco systems, Compaq and Intel.

RosettaNet Framework [15] consists of Partner Interface Processes (PIPs), a master dictionary and an implementation framework, the relationship among which can be expressed with the following analogy: RosettaNet dictionaries provide the words, the RosettaNet Implementation Framework (RNIF) acts as the grammar, and RosettaNet Partner Interface Processes (PIP) form the dialog.

RosettaNet dictionaries provide a common vocabulary platform for conducting business within the trading network:

- The *RosettaNet Business Dictionary* contains information about the trading partners like Business Properties (e.g. business address), Business Data Entities (like ActionIdentity), and Fundemental Business Data Entities (e.g. BusinessTaxIdentifier, AccountNumber). There is only one business dictionary that encompasses all supply chains like Electronic Components (EC), Information Technology (IT), etc.

- The *RosettaNet Technical Dictionary (RTD)* provides properties for describing products and services. Note that the RTD integrated the two formerly distinct dictionaries, namely *EC Technical Dictionary* and *IT Technical Dictionary*.

The RosettaNet framework enables supply chain business partners to execute interoperable electronic business (e-business) processes by developing and maintaining PIP implementation guidelines. RosettaNet distributes PIPs to the trading partners, who use these guidelines as a road map to develop their own software applications. PIPs include all business logic, message flow, and message contents to enable alignment of two processes.

In order to do electronic business within the RosettaNet framework, there are a number of steps the partners have to go through. First, the supply chain partners come together and analyze their common inter-company business scenarios (i.e., public processes), that is, how they interact to do business with each other, which documents they exchange and in what sequence. These inter-company processes are in fact, the "as-is" scenarios of their way of doing business with each other. Then they re-engineer these processes to define the electronic processes to be implemented within the scope of the RosettaNet Framework.

An electronic business process includes both the interactions between partner companies, and the private processes within the company. The interactions between supply chain partners are analyzed to create RosettaNet's Partner Interface Processes (PIPs). Note that each partner implements its own private processes. RosettaNet provides guidelines

Business Document	Description
Purchase Order Request	A request to accept a purchase order for fulfillment
Purchase Order Confirmation	Formally confirms the status of line item(s) in a Purchase Order. A Purchase Order line item may have one of the follwing states: accepted, rejected, or pending.

Figure 4: RosettaNet Business Documents for PIP3A4

nly for PIPs which are the public part of the inter-company rocesses.

To have a manageable framework in developing PIPs, tosettaNet has grouped supply chain processes into clus-ers, which are further grouped into segments. For example,)luster 3 is "Order Management" and "Segment 3A" in this luster is about "Quote and Order Entry". As an example if the PIPs in this segment, "PIP3A4: Manage Purchase)rder" has the purpose of supporting a process between rading partners that involves issuing a purchase order and icknowledging that purchase order. This PIP also supports he capability to cancel or change the purchase order based on the acknowledgement response.

The choreography, that is, the sequence of steps within a PIP is given in the "blueprint" which is a business pro-cess flow diagram. The blueprint for "PIP3A4: Manage Purchase Order" is given in Figure 2. A number of tables provide accompanying information as presented in Figures 3, 4, 5, and 6. The information provided in these figures are used in mapping PIPs to ebXML "Binary Collaborations" as explained in Section 3.

2.2 UDDI

UDDI [16] is jointly proposed by IBM, Microsoft and Ariba. It is a service registry architecture that presents a standard way for businesses to build a registry, discover each other, and describe how to interact over the Internet. Currently IBM and Microsoft are running public registries [17] and Hewlett-Packard is expected to launch a third one.

The UDDI information model, defined through an XML schema, identifies five core types of information. These core types are business, service, binding, service specifications information and relationship information between two parties. Through these data structures, business entities describe information about businesses like their name, description, services offered and contact data. Business services provide more detail on each service being offered. Services can then have multiple binding templates, each describing a technical entry point for a service (e.g., mailto, http, ftp, phone, etc.). These structures use category bags for categorization purposes. An item in a category bag contains a tModel key and an associated OverviewDoc element.

tModels provide the ability to describe compliance with a specification, a concept, or a shared design. When a particular specification is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification. The specification is not included in the tModel itself. The "OverviewDoc" and "OverviewURL" elements of tModels are used to point at the actual source of a specification. More precisely, the use of tModels in UDDI is two-fold:

- *Defining the technical fingerprint of services:* The primary role that a tModel plays is to represent a technical specification on how to invoke a registered service, providing information on the data being exchanged, the sequence of messages for an operation and the location of the service. Examples of such technical specifications include Web Services Description Language (WSDL) [20] descriptions.

- *Providing abstract namespace references:* In UDDI, businesses, services and tModels can specify the categories to which they belong in their category bags. Categorization facilitates to locate businesses and services by relating them to some well-known industry, product or geographic categorization code set. Currently UDDI uses the North American Industrial Classification Scheme (NAICS) [10] taxonomy for describing what a business does; the Universal Standard Products and Services Classification (UNSPSC) [19] for describing products and services offered; and ISO 3166, a geographical taxonomy for determining where a business is located. It should be noted that any number of categories can be referenced in category bags.

The functionality provided by UDDI can be summarized as follows:

- It is possible to locate businesses and their services by their names published in the UDDI registry.

- The categories referenced in the category bags can be used to find businesses or services of a particular category. For example a user looking for a service for a particular product type can first obtain the product code from one of the defined taxonomies, like NAICS [10] or UNSPSC [19]. Assuming that the user wants to access the services related with optical computer disks, he obtains the UNSPSC code of "Magneto optical disks" which is "43.18.16.07.00" and searches the UDDI registry by using the APIs provided to find the businesses and their services that contain this code in their category bags. However if a business fails to provide this exact code in its category bag, it becomes impossible to locate it in this way.

- UDDI expresses the compliance of businesses and services that reference the same tModel in their descriptions.

The functionality of an UDDI registry can be improved by describing the semantics of Web Services. A comprehensive semantic framework improving service discovery through UDDI is presented in [3].

PARTNER ROLE DESCRIPTION		
Role Name	Role Description	Role Type
Buyer	An employee or organization that buys products for a partner type in the supply chain	Functional
Seller	An organization that sells products to partners in the supply chain	Organizational

Figure 5: Partner Role Descriptions for PIP3A4

Role Name	Activity Name	Acknowledgment of Receipt		Time to Acknowledge Acceptanace	Time to Perform	Retry Count	Is Authorization Required?	Non-Repudiation of Origin and Content?
		Non-Repudiation Required?	Time to Acknowledge					
Buyer	Request Purchase Order	Y	2 hrs	N/A	24 hrs	3	Y	Y

Figure 6: Business Activity Performance Controls for PIP3A4

2.3 ebXML

ebXML [4] is an initiative from OASIS [11] and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [18]. ebXML aims to provide the exchange of electronic business data in Business-to-Business (B2B) and Business-to-Customer (B2C) environments. The vision of ebXML is to create a single set of internationally agreed upon technical specification that consists of common XML semantics and related document structures to facilitate global trade. It should be noted that ebXML is not about creating standard schemes or DTDs for common business documents such as purchase orders or invoices, but instead is about creating an infrastructure.

The ebXML architecture specifies the following functional components:

- *Trading Partner Information:* The Collaboration Protocol Profile (CPP) provides the definition (DTD and W3C XML Schema) of an XML document that specifies the details of how an organization is able to conduct business electronically. It specifies such items as how to locate contact and other information about the organization, the types of network and file transport protocols it uses, network addresses, security implementations, and how it does business (a reference to a Business Process Specification). The Collaboration Protocol Agreement (or CPA) specifies the details of how two organizations have agreed to conduct business electronically. It is formed by combining the CPPs of the two organizations.

- *Business Process Specification Schema (BPSS):* The Business Process Specification Schema [8] provides the definition of an XML document (in the form of an XML DTD) that describes how an organization conducts its business. While the CPA/CPP deals with the technical aspects of how to conduct business electronically, the Specification Schema deals with the actual business process. The process specification document defines, among other things, the request and response messages for each business transaction and the order in which the business transactions should occur.

- *Messaging Service:* ebXML messaging service [7] provides a standard way to exchange messages between organizations reliably and securely. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP.

- *Core Components:* ebXML provides a core component architecture where a core component is a general building block that basically can be used to form business documents.

- *Registry/Repository:* A registry is a mechanism where business documents and relevant metadata can be registered such that a pointer to their location, and their metadata can be retrieved as a result of a query. A registry can be established by an industry group or standards organization. A repository is a location (or a set of distributed locations) where a document pointed at by the registry reside and can be retrieved by conventional means (e.g., http or ftp). An ebXML Registry [5, 9] provides a set of services that manage the repository and enable the sharing of information between trading partners. Note that business processes, CPPs, business document desciptions and core components are published and retrieved via ebXML Registry Services. A trading partner may discover other trading partners by searching for the CPPs in the registry. The ebXML Messaging Service is used as the transport mechanism for all communication into and out of

the Registry.

The ebXML infrastructure is modular, and with few exptions these infrastructure components may be used someıat independently; they are loosely related. The elements the infrastructure may interact with each other, but in ost cases are not required to. The CPP, CPA and Busi:ss Process Specifications may be stored in an ebXML ›mpliant Registry, but this is not required. An ebXML ›mpliant Registry may store any type of object, including ›n-XML objects. However, all communications with the :gistry must use the ebXML messaging service [14].

. MAPPING ROSETTANET PIPS INTO EBXML BUSINESS PROCESSES

Both the RosettaNet PIPs and the ebXML Binary Colıborations define dialogs between e-Business partners and ıclude the following information:

- the sequence of steps required to execute an atomic business process between two trading partners
- the activities involved
- the roles of the partners
- the specification of structure and content of the business documents exchanged
- the security, authentication, time and performance constraints on the interactions.

In other words RosettaNet PIPs correspond to ebXML Binary Collaborations. The difference is on how this information is expressed. In RosettaNet, PIPs are defined in Unified Modelling Language (UML) and in ebXML, Binary Collaborations are XML documents conforming to given XML DTDs. Mapping RosettaNet PIPs into ebXML Binary Collaborations is straight forward and can easily be automated by mapping the UML output to an XML document through XMI [21]. However this can also be done manually. In this section we describe this process through an example.

Mapping a RosettaNet PIP into an ebXML "Binary Collaboration" implies filling in the template XML document (with a given ebXML DTD) from RosettaNet Business Process Flow Diagrams and associated Tables for a PIP.

As an example, consider the XML document given in Figure 7. This is an ebXML Binary Collaboration document conforming to ebXML Business Process Specification DTD. This document expresses the semantics of RosettaNet PIP3A4, namely, Request Purchase Order. In this figure, the BusinessTransaction name in line 1 is obtained from the corresponding RosettaNet Business Process Flow Digram given in Figure 2. The values of attributes given in line 2 and line 5, namely, *isNonRepudiationRequired* and *timeToAcknowledge receipt* are available from Table 3 which provides the message exchange controls for "RequestPurchaseOrder" PIP in RosettaNet. The *businessDocument* names in lines 3 and 6 are obtained from Table 4 which describes the Business Documents for the PIP in question. The BinaryCollaboration name given in line 9 is the name of the PIP and the *timeToPerform* attribute is obtained from Table 6 describing Business Activity Performance controls. The roles given in lines 10 and 11 are available from Table 5 which describes "Partner Role Descriptions" for this PIP. Finally, the information in lines 12 through 15 is again available form Business Process Flow Digram given in Figure 2.

It is clear that creating an ebXML "Binary Collaboration" corresponding to RosettaNet PIP involves filling in the ebXML business process template with data obtained from the corresponding RosettaNet PIP which is a straight forward task. As mentioned earlier it is also possible to automate this task by directly mapping RosettaNet PIPs expressed in UML to XML through [21] by using tools such as the one given in [12].

4. GUI TOOL FOR BUILDING EBXML BUSINESS PROCESS SPECIFICATIONS FROM BINARY COLLABORATIONS

A business process in ebXML describes how trading partners take on roles, relationships and responsibilities to facilitate interaction with other trading partners. The interaction between roles takes place as a choreographed set of "Binary Collaborations".

Consider, for example, a scenario where a buyer requests the price and availability of some products from a seller (PIP3A2). After receiving the response, the buyer initiates a Purchase Order Request (PIP3A4). The seller, on the other hand, after acknowledging the Purchase Order Request, sends an invoice notification (PIP3C3) to the buyer. There is a third party in this scenario, which is a shipper. The seller sends a transportation request (PIP3B1) to the shipper. The shipper, after shipment of the goods, sends the status of the shipment (PIP3B3). When buyer receives the shipment, it sends a shipment receipt notification (PIP4B2) to the seller. Finally, the seller prepares a billing statement and notifies the buyer (PIP3C5).

To facilitate the process of building such complex ebXML business scenarios, we have developed a GUI tool which makes it possible to graphically construct ebXML business processes specifications from Binary Collaborations obtained through RosettaNet PIPs. Figure 8 shows how the described scenario is defined through the GUI tool.

The tool allows the user to select PIPs; drag them onto the canvas and draw the transitions among them. After a PIP is placed onto the canvas the user specifies one of the defined roles as the InitiatingRole in the PIP. A condition expression in XPath language may be inserted for the transitions from one PIP to the other. This expression forms the *ConditionExpression* element which is checked to decide whether Business Transaction Activity will be executed. Another useful attribute of the PIP on the canvas is the URLs of the Binary Collaborations. This attribute is also assigned by the user.

5. REGISTERING EBXML CPPS TO UDDI REGISTRY

To facilitate the process of conducting eBusiness, potential trading partners need a mechanism to publish information about the business processes they support along with specific technology implementation details about their capabilities for exchanging business information. In ebXML, this information is available through the Collaboration Protocol Profile (CPP). The CPP contains essential information about the trading partner like contact information, indus-

```
BusinessTransaction name="REQUEST_PurchaseOrder">
 <RequestingBusinessActivity isNonRepudiationRequired="true" timetoAcknowledgeReceipt="PT2H">
    <DocumentEnvelope businessDocument='//BusinessDocument [@name="PurchaseOrderRequest"]'/>
 </RequestingBusinessActivity>
<RespondingBusinessActivity isNonRepudiationRequired="true" timeToAcknoewledgeReceipt="PT2H">
    <DocumentEnvelope businessDocument='//BusinessDocument [@name="PurchaseOrderConfirmation"]'/>
</RespondingBusinessActivity>
:/BusinessTransaction>

<BinaryCollaboration name=REQUEST_PurchaseOrder" timeToPerform= "PT24H">
 <InitiatingRole name="Buyer"/>
 <RespondingRole name="Seller"/>
 <BusinessTansactionActivity name= "RequestPurchaseOrder" businessTransaction '
                  // businessTransaction[@name= REQUEST_PurcahseOrder"]'
                  fromAuthorizedRole='../Buyer" toAuthorizedRole="../Seller"/>
                  fromAuthorizedRole='../Buyer" toAuthorizedRole="../Seller"]'/>
 <Start toBusinessState='../BusinessTransactionActivity [@name= "RequestPurchaseOrder"]'
 <Success fromBusinessState='../BusinessTransactionActivity [@name= "RequestPurchaseOrder"]'
          guardCondition="Success" guardExpression='//PurchaseOrderConfirmation/
                                                    GlobalDocumantationCode="Accept"/>

 <Failure fromBusinessState='../BusinessTransactionActivity[@name= RequestPurchaseOrder"]'
          guardCondition="BusinessFailure" guardExpression='//PurchaseOrderconfirmation/
          GlobalDocumentCode='Reject"'/>
 <BinaryCollaboration>
```

Figure 7: An Example ebXML Binary Collaboration Document

, classification, supported Business Processes, Interface quirements, etc.

In the infrastructure developed, ebXML compliant doc- nents are registered to UDDI. While registering CPPs UDDI registry, a mechanism is necessary to facilitate eir discovery. Currently there is "uddi-org:types" taxon- ay where, for example, Web Services Description Language VSDL) is classified as "wsdlSpec". It would help to dis- ver CPPs whose business process specifications are based RosettaNet PIPs if RosettaNet is classified with "uddi- g:types" taxonomy like WSDL.

In our implementation we have defined tModels for each the PIPs to help with their discovery. By deveoping a Model for each PIP and including the keys of these tModels the category bags of the related CPPs, we facilitate the iscovery of CPPs referencing business processes conforming RosettaNet PIPs.

Other documents registered at UDDI include the business ocuments exchanged by the parties. The business docu- nents we use in our implementation are based on the DTDs rovided by RosettaNet. However, it is possible to use a ommon document library like Universal Business Language UBL) as announced by OASIS when it becomes available n the future.

5. AN AUTOMATED TOOL FOR CONFIGURING THE CPA GIVEN CPPS

A Collaboration Protocol Agreement (CPA) describes the *Messaging Service* and the *Business Process* requirements that are agreed upon by both partners. The intent of the CPA is to provide a high-level specification that can be easily comprehended by humans and yet is precise enough for enforcement by computers.

The information in the CPA is used to implement Business Service Interfaces (BSI) to enable exchange of messages with trading parties. The ebXML *Message Service Handler* specification is used to implement the exchange of messages.

ebXML provides the guidelines to generate a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs) which involves the following [6]:

- Matching business processes and the roles
- Matching transport and transport security
- Matching document packaging and document security

6.1 Matching Business Processes and the Roles

As an initial requirement to creating a CPA, the *Process Specification* elements of the two parties must match. That is, both of the processes should contain the same binary collaborations and in the same choreography. In our framework, the binary collaborations are obtained from RosettaNet PIPs and each business process definition in the UDDI registry contains tModels corresponding to RosettaNet PIPs. Therefore, finding out the business processes that contain the same binary collaborations is easy: it is enough to check that they have the same tModel keys. In order to check whether the choreography of these binary collaborations are the same; it is necessary to match the transitions between Binary Collaborations as specified in Multi-party Collaboration part of the business process.

Matching the roles in two CPP implies checking whether the roles of the partners given in the CPPs are complementary. "PartyInfo" element in a CPP contains a subtree of elements called "CollaborationRole". These sets in the CPPs are compared to find out whether the roles are complementary within the specified *BusinessCollaboration*. As an example, the roles "buyer" and "seller" are complementary in a "Request Purchase Order" business process specification.

6.2 Matching Transport and Transport Security

Matching transport means matching the SendingProtocol capabilities of one party with the ReceivingProtocol capabilities of the other party in a complementary role. The CPP DTD (or Schema) has a ServiceBinding element that

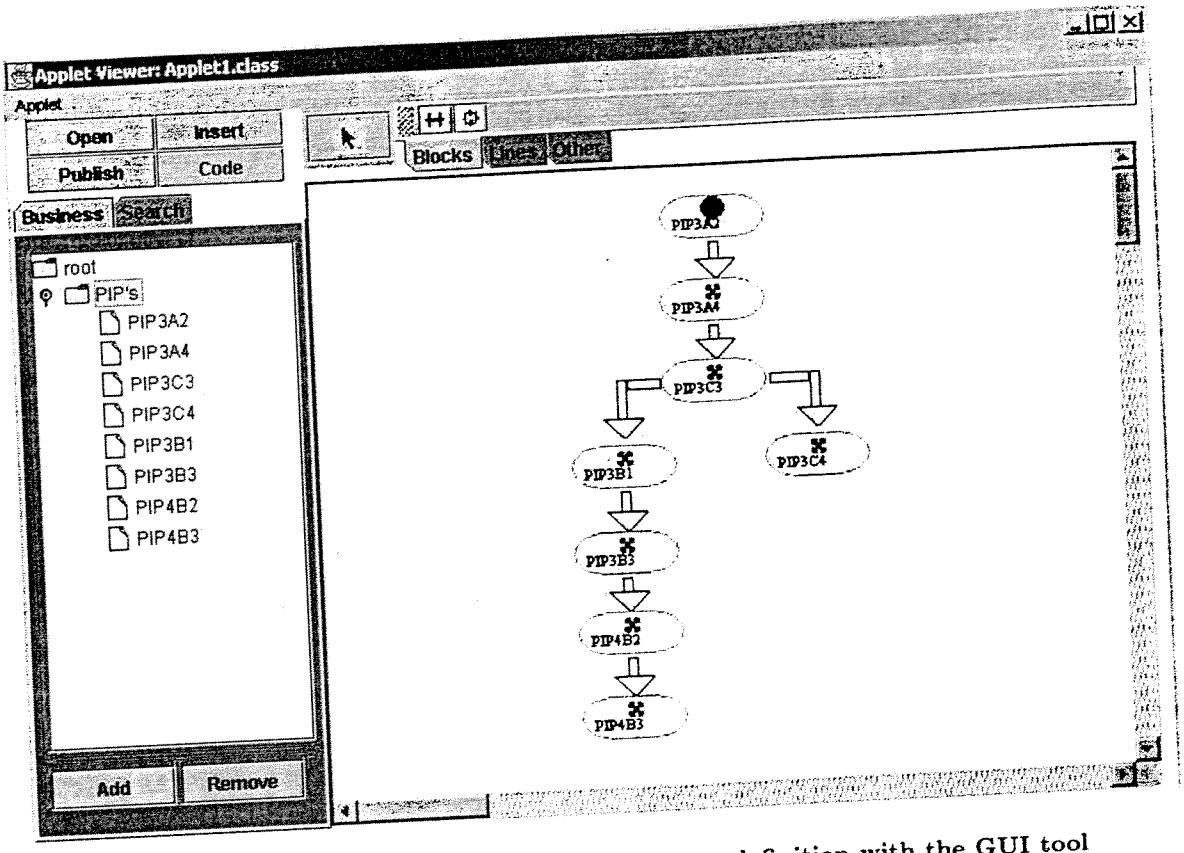


Figure 8: An example ebXML business process definition with the GUI tool

oints to the relevant information through the "channelId" ttribute. "channelId" attribute's value defines the Deliv-ryChannels within each CPP. The DeliveryChannel has a ransportId attribute that specifies the relevant Transport ubtrees.

As an example, suppose that a buyer's CPP has the fol-owing Transport entry:

```
<Transport transportId = "buyerid001">
   <SendingProtocol>HTTP</SendingProtocol>
   <ReceivingProtocol> FTP </ReceivingProtocol>
   <Endpoint uri = "https://www.buyername.com/
            po-response" type = "allPurpose"/>
   <TransportSecurity>
      <Protocol version = "1.0">TLS</Protocol>
      <CertificateRef certId =  certid001">BuyerName
      </CertificateRef>
   </TransportSecurity>
</Transport>
```

Assume further that a seller's CPP has the following Transport entry:

```
<Transport transportId = "sellid001">
   <SendingProtocol>FTP</SendingProtocol>
   <ReceivingProtocol> HTTP </ReceivingProtocol>
   <Endpoint uri = "https://www.sellername.com/
                os_here" type = "allPurpose"/>
   <TransportSecurity>
      <Protocol version = "3.0">SSL</Protocol>
      <CertificateRef certId ="certid002">Sellername
      </CertificateRef>
   </TransportSecurity>
</Transport>
```

It is clear from the example that the transport elements of the two CPPs match since the "SendingProtocol" and the "ReceivingProtocol" in the complementary roles match; one is HTTP and the other is FTP. If such a match can not be found, then the available exception handling mechanisms are activated.

Matching transport security involves finding an agreement between the versions and the values of the security protocols specified in the "Protocol" element. To facilitate this process, we provide a Transport Security lookup table that keeps track of compatible protocols and versions. For example if SSL-3 and TLS-1 are defined to be compatible protocols, the examples given above will result in a match in transport security.

6.3 Matching Document Packaging and Document-Level Security

In ebXML messaging structure, message payloads are packaged using the MIME multipart/related content type. MIME is used as a packaging solution because of the diverse nature of information exchanged between partners in eBusiness environments. For example, a complex business transaction between two or more trading partners might require a payload that contains an array of business documents (XML or other document formats), binary images, or other related business information. And due to this possible complexity of the document packaging, checking matches is not a straight forward process. The need to check the document-level security of the involved documents where each might have a radically different security mechanism adds to the

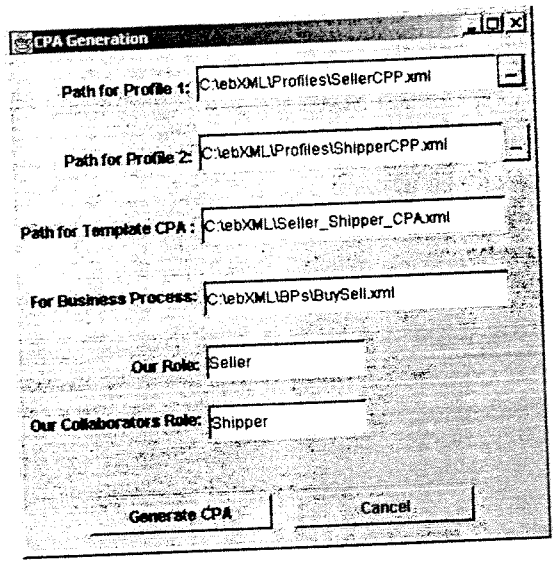


Figure 9: A Screenshot from CPA Generation Tool

complexity of the problem.

"ServiceBindings" elements in CPPs contain the packaging structure. The CPA tool provided first tries to match the simple parts in the packaging and then proceeds with "CompositeList". For matching the document level security, the tool seeks for human assistance due to the complexity involved in this process.

Figure 9 demonstrates a screenshot from the GUI tool developed for automating CPA generation. Further details of this work are given in [13].

7. A B2B SERVER IMPLEMENTATION FOR EBXML BUSINESS PROCESS SUPPORT

For businesses to become ebXML compliant, they should be able to process incoming ebXML messages. This necessitates a software component at the site of the ebXML compliant partner for handling ebXML messages.

Furthermore, UDDI provides a registry mechanism mainly for describing and invoking Web services. In contrast, ebXML, through CPPs, point to the public business processes. The main difference between a service and a business process is as follows: A service is an atomic unit of processing; it is invoked either by passing a message or invoking one or more Remote Procedure Calls (RPCs). Whereas a business process involves collaboration and hence exchange of a sequence of messages according to a predefined choreography of interactions with well defined roles for the involved partners. In ebXML, the atomic unit of interaction between the trading partners is a "Binary Collaboration", and an ebXML public business process may involve more than one "Binary Collaboration" and more than two parties. That is, multi party collaboration is possible.

Therefore, although UDDI registry can be used to discover ebXML CPPs, further mechanisms are necessary at ebXML compliant partners' sites to keep track of the message traffic.

Hence, providing an ebXML infrastructure implies providing mechanisms to ebXML trading partners for properly processing ebXML messages as well as keeping track of the data and control flow in the ebXML business processes. We developed a generic B2B Server implementation for this purpose. The main functionalities of the server are to implement an ebXML Messaging Service infrastructure and to provide workflow capabilities to keep track of business processes.

7.1 ebXML Messaging Service Implementation

An ebXML Message Service mechanism [7] is implemented as a component of the B2B server. An ebXML message consists of an optional transport protocol specific outer envelope which contains a protocol independent ebXML Message Envelope as its payload. The ebXML Message Envelope is packaged using the MIME multipart/related content type.

The ebXML Message Service may be conceptually broken down into following three parts: (1) an abstract Service Interface, (2) functions provided by the Message Service Handler (MSH), and (3) the mapping to underlying transport service(s).

The ebXML message processing involves the following:

- Header Processing - the creation of the SOAP Header elements for the ebXML message.
- Header Parsing - extracting or transforming information from a received SOAP Header or Body element into a form that is suitable for processing by the MSH implementation.
- Security Services - digital signature creation and verification, authentication and authorization.
- Reliable Messaging Services - handles the delivery and acknowledgment of ebXML messages sent with delivery semantics of "Once And Only Once".
- Message Packaging - the final enveloping of an ebXML Message (SOAP Header or Body elements and payload) into its SOAP Messages with Attachments container.

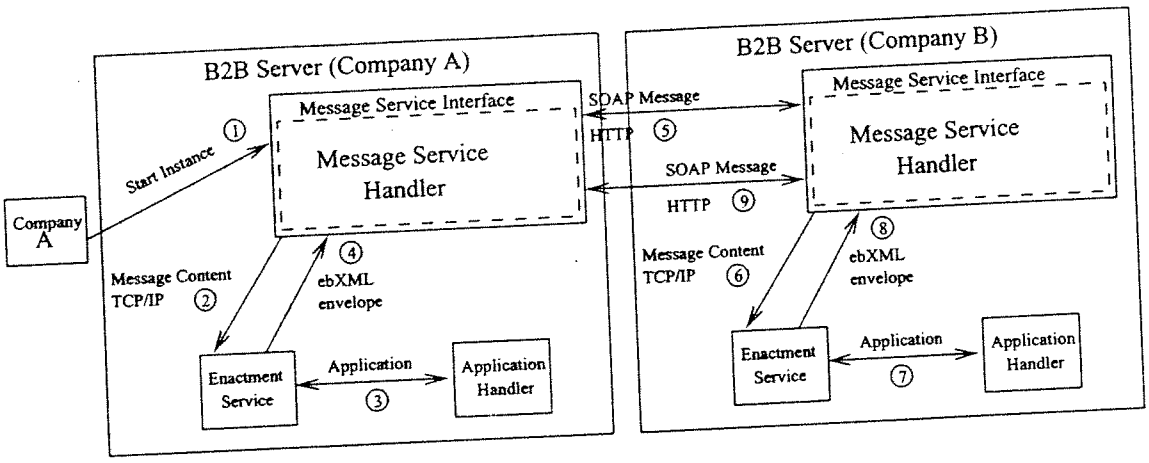


Figure 10: Architecture of the B2B Server

- Error Handling - this component handles the reporting of errors encountered during MSH or Application processing of a message.
- Message Service Interface - an abstract service interface that applications use to interact with the MSH to send and receive messages and which the MSH uses to interface with applications that handle received messages.

The Message Service Handler is implemented in Java using Apache SOAP library. Apache SOAP implementation provides for adding MIME attachments to the SOAP messages. Messaging Service Implementation provides an API to facilitate preparing ebXML messages. The API includes interfaces to form the fields of an ebXML message, to construct the envelope of the message and to prepare a SOAP message. The software developed also provides a transport mechanism to send the prepared message to the destination according to the related protocol (e.g. HTTP, SMTP, FTP). It uses the corresponding protocols of SOAP messaging service for this purpose. Note that the transport protocol to be used is specified in the CPA.

The Message Service Interface implemented interacts with the Message Service Handler to send, receive and route the messages. The routing information, that is, locating the software to process the message, is obtained from the "eb:Service" element in the ebXML message header. Note that the "eb:Service" element is originally contained in the CPA. Since the B2B server handles all the message traffic and locates the applications to be invoked, the type attribute of "eb:Service" element in the CPAs using this infrastructure should be "b2bserver".

7.2 Workflow Enactment Service

Enactment service component of the B2B server handles the CPA and the ebXML business processes running at trading partners' sites. Note that the CPA and the process specification document that it references define a conversation between two parties. The conversation represents a single unit of business as defined by the "Binary Collaboration" component of the Process-Specification document. The conversation consists of one or more "Business Transactions", each of which is a request message from one party and zero or one response message from the other party.

The initiating party in each business process starts an instance of the business process by sending a special message to the B2B server. This message includes the business process id, type (e.g. binary or multiparty) and the name of the collaboration to be executed. Upon receiving this message, B2B server activates the enactment service, and the process instance is started. Enactment service assigns a unique id and a conversation id to each business process instance created. ConversationId together with the CPAId, uniquely identifies the instance at all the sites involed.

As described in Section 7.1, B2B server receives messages from other parties through ebXML Message Service Interface (MSI) which is a component of Message Service Handler (MSH). MSH in turn, parses the ebXML message, and sends it to Enactment Service. Enactment service locates the related business process instance and its state using the information provided in the message.

7.3 How the System Works

The architecture of B2BServer is shown in Figure 10. Assuming that Company A has the InitiatingRole in the business process, an instance of a business process is started by sending a special message to the B2B Server of Company A. The ebXML Message Service Handler (MSH) component of the B2B Server receives the message. MSH decomposes the message into its parts (e.g. message header, message body, attachments, etc.) and sends these parts to the Enactment Service. Enactment Service then locates the related business process and starts a new instance.

It should be noted that an electronic business process includes both the interactions between partner companies, and the private processes within the company. Each party executes its own internal processes and interfaces them with the Business Collaboration as described by the CPA and Process Specification documents. The execution of a private business process may involve activating some internal processes (Business Transaction Activity, BTA, in ebXML terminology) at the trading partner's site. To automate the invocation of Business Transaction Activities, the B2B server provides a mechanism through its configuration file to assign applications to the BTAs specified in business processes.

After the completion of internal private processes, the Enactment Service sends a message to the B2B Server of the other party to inform the result. To send a message, Enact-

t Service contacts Message Service Interface (MSI) by ling an ebXML envelope prepared using the messaging provided. MSI then sends this message to the specified y using SOAP protocol. Finally the Enactment Service ks the transitions of the related business process and eeds to the next state (if any). Company B, on the r hand, has the responding role and processes the mes- s it receives according to the process definition exactly he same way.

CONCLUSIONS

n this paper, a *hybrid* B2B infrastructure is described t exploits the strengths of some of the existing standards, nely ebXML, UDDI and RosettaNet. An UDDI registry ised to store ebXML documents including ebXML busi- s processes that are obtained through RosettaNet PIPs. number of add-on services to facilitate B2B e-commerce also provided by the system developed, like a GUI to istruct ebXML compliant business processes; an enact- nt service to keep track of data and control flow in the ecution of these processes; a tool to assist the derivation CPAs given two CPPs; and an ebXML messaging infras- icture.

A prototype of the system developed within e scope of the ebXML project is available from ttp://www.srdc.metu.edu.tr/ebXML" upon request. ie prototype is implemented in Java JDK 1.2. Xerces ML parser is used to parse necessary XML documents. karta Tom Cat servlet Engine is used for Servlet related erations. Apache SOAP, javamail-1.2 and jaf-1.0.1 are ed together to implement the ebXML Messaging Service. sual Cafe is used to implement the GUI tool for Building XML business processes.

Our future work includes developing an ebXML compliant gistery and repository [2].

REFERENCES

] Bussler, C., "B2B Protocol Standards and their Role in Semantic B2B Integration Engines", IEEE Bulletin of the TC on Data Engineering, http://www.research.microsoft.com/research/db/debull/issues-list.htm, Vol. 24, No. 1, March 2001.

2] Carkacioglu, L., "Implementation of an ebXML Compliant Registery", MSThesis, Middle East Technical University, in preparation.

3] Dogac, A., Cingil, I., Tambag, Y., Laleci, G.B., Kabak, Y., "Describing the Semantics of Web Services and UDDI", submitted for publication.

4] ebXML, http://www.ebxml.org/

5] ebXML Registry Services Specification v1.0, http://www.ebxml.org/specs/ebiRS.pdf, May 2001.

6] ebXML Collaboration Protocol Profile and Agreement Specification, http://www. ebxml.org/specs/ebCCP.pdf.

7] ebXML Message Service Specification v1.0, http://www.ebxml.org/specs/ebMS.pdf, May 2001.

8] ebXML Business Process Specification Schema v1.0, http://www.ebxml.org/specs/ebBPSS.pdf, May 2001.

9] ebXML Registry Information Model v1.0, http://www.ebxml.org/specs/ebRIM.pdf, May 2001.

10] North American Industrial Classification Scheme (NAICS) codes http://www.naics. com.

[11] OASIS, http://www.oasis-open.org/cover/siteIndex.html

[12] Objecteering, http://www.objecteering.com/us/produits_pe.htm

[13] Pembecioglu, P., "A Tool for Automating CPA Construction in ebXML Compliant Registries", MSThesis, Middle East Technical University, in preparation.

[14] Rawlins, M., "Overview of the ebXML architectures", http://rawlinsecconsulting.com/

[15] RosettaNet, http://www.rosettanet.org/

[16] UDDI: Universal Description, Discovery and Integration, http://www.uddi.org, 2001.

[17] UDDI Registry, http://www-3.ibm.com/services/uddi/

[18] UN/CEFACT, http://www.diffuse.org/fora.html#CEFACT

[19] Universal Standard Products and Services Classification (UNSPSC), http://eccma.org/unspsc

[20] Web Service Description Language (WSDL), http://www.w3.org/TR/wsdl

[21] XML Metadata Interchange (XMI), http://www-4.ibm.com/software/ad/library/standards/xmi.html

A PLATFORM FOR SEMANTICALLY ENRICHED MOBILE SERVICES

Asuman Dogac, Gokce Laleci, Yildiray Kabak, Gokhan Kurt, Aybar Acar

Software R&D Center, Dept. of Computer Eng. Middle East Technical Univ., 06531, Ankara, Turkey
Contact author: Asuman Dogac, asuman@srdc.metu.edu.tr
Phone: +90-312-2105598, Fax: +90-312-2101004

ABSTRACT

When looking towards the future of web-services for mobile devices, it is predicted that the breakthrough will come when the agents turn the entire collective of web-services from the existing dormant mass of information the users need to surf, browse and discover, into a dynamic set of capabilities serving the mobile user in fulfilling her personal needs according to her profile and context. This breakthrough involves the following challenges: first, the Web service discovery and execution need to be automated to bring the ultimate ease-of-use to the end users. This requires describing the semantics of Web services and also capturing the context within which the user operates, not for example, just her location. To provide the essential ease of use, the intelligent agents are needed to discover, compose and execute Web services according to user needs, preferences and context. Finally, to make Web services accessible to mobile devices through software agents effectively, their limitations have to be taken into consideration. We describe an infrastructure to make semantically described Web services available to mobile devices through agent technology and service registries. We will identify the need and the relevance of the involved standards and investigate the open research problems.

1. INTRODUCTION

While the bandwidth and processing power of mobile devices continue to increase, presenting the users with seemingly limitless opportunities, the lack of semantics on the Web has become the bottleneck. For agents to discover Web services according to their functionality, the metadata describing the functionality of the services should be provided. Without metadata, it is difficult to automate the discovery of services according to their functionality or properties, to relate services to product instances, or to compose and monitor them. Recently there is an important initiative in describing Web service semantics, namely, DAML-S [7]. DAML-S is a comprehensive effort based on DAML+OIL [6]. However the DAML-S specification, which defines an upper ontology for services, is not complete yet.

Once the Web service semantics are available, the software agents can free humans from a number of chores imposed by the contemporary Internet. In fact agents represent the great opportunity towards a new and completely different computing model for Internet access using mobile devices. In particular, agents will turn the Web services into proactive entities serving the end-user according to their needs, preferences and contexts. Intelligent User Interface Agents that are aware of the mobile users' profiles as well as their context, are capable of intelligently orchestrating the Web services to serve the user's needs. With context we mean user's state, her social setting, and spatial-temporal situation [28]. The privacy issues can be handled by using the World Wide Web Consortium's Platform for Privacy Preferences Standard [23].

Another important issue in agent infrastructures is interoperability, which has been properly addressed through a number of standardization efforts like MASIF [18] and FIPA [10]. There are numerous agent development platforms, like ZEUS [45], JADE [14], and LEAP [17] with various degrees of abstraction and completeness complying with the mentioned standards. Currently, the Agentcities [2] project is creating a network where agents running on different platforms, owned by different organizations, implemented in different ways and providing diverse services, can interact. Although such efforts provide an interoperability framework for agents such as a common language for agents, agents that communicate in a common language will still be unable to understand each other if they use different vocabularies for representing shared domain concepts. Therefore there is also a need for ontology interoperation. FIPA supports ontology interoperation through an Ontology Agent, which communicates with ontology servers through OKBC (Open KnowledgeBase Connectivity). OKBC is based on KIF and needs to be extended to support the sophisticated classification and resource properties introduced by DAML+OIL ontologies.

Another issue to be addressed is the following: Even when the semantics are well defined, the discovery of the Web services should be left to search agents only as the last resort. Instead, domain specific registries should be used to gain search efficiency. There are a number of registries developed as a result of industrial initiatives for service discovery such as UDDI [32] and ebXML [8]. Defining the semantics of services is not enough to discover them through registries; there should be a mechanism in the registry to relate the semantics defined with the service advertised. In this respect UDDI registries define only tModels to describe compliance with a specification. However ebXML provides constructs to define meta-data in the registry and to relate these meta-data elements with registry objects.

In this paper we propose a platform for semantically enriched mobile services to address these issues. The paper is organized as follows: Section 2 introduces the semantics of Web services and briefly describes DAML-S. Section 3 summarizes the FIPA compliant agent platform that we will use in our implementation, namely, JADE. Another FIPA compliant agent for mobile devices, namely, LEAP is also mentioned in this section. Section 4 describes two main service discovery platforms, UDDI and ebXML. In Section 5, we describe how to develop ontologies in DAML+OIL to be used in FIPA compliant platforms by extending OKBC. Section 6 presents personalization of the Web services for mobile users by respecting their rights to privacy. Section 7 describes the intelligent user interface agent and the service orchestration tool. In Section 8, we introduce the constraints and requirements of mobile devices. Finally, Section 9 gives an overall view of the system.

2. SEMANTICS OF WEB SERVICES

Recently, efforts in describing the semantics of the Web have gained momentum. The next-generation Web, which is called the "Semantic Web" [1], aims at providing access to structured collections of information and sets of inference rules that the computers can use to conduct automated reasoning. An important effort in this respect is DAML+OIL [6], a language for expressing sophisticated classifications and properties of resources. The Semantic Web will enable greater access not only to content but also to services on the Web and there are several efforts for describing the semantics of Web services. Among these DAML-S [7] is a comprehensive effort.

The top-level class in DAML-S service taxonomy is the class *Service* that has the following three properties [7]:

- Service Profile: The service profile defines "what the service does"; that is, it gives the type of information needed by a service-seeking agent to determine whether the service meets its needs (typically such things as input and output types, preconditions and post conditions, and binding patterns). Service profiles consist of three types of information: a human readable description of the service; a specification of the functionalities that are provided by the

service; and a host of functional attributes which provide additional information and requirements of the service that assist when reasoning about several services with similar capabilities. While service providers use the service profile to advertise their services, service requesters use the profile to specify what services they need and what they expect from such a service. For instance, a service seeking agent may look for a service selling an ``IBM hard disk" that is second-hand. The role of the registries is to match the request against the profiles advertised by other services and identify which services provide the best match.

- Service Model: The service model defines "how the service works"; that is, it describes what happens when the service is carried out. For non-trivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways: (1) to perform a more in-depth analysis of whether the service meets its needs; (2) to compose service descriptions from multiple services to perform a specific task; (3) during the course of the service enactment, to coordinate the activities of the different participants; (4) to monitor the execution of the service.

- Service Grounding: Service grounding specifies the details of how an agent can access a service. Typically service grounding will specify a communications protocol (e.g., RPC, HTTP-FORM, CORBA IDL, SOAP, Java RMI), and service-specific details such as port numbers used in contacting the service.

- DAML-S also provides mechanisms to model processes. The two major components of a process model are the *process model*, which describes a service in terms of its component actions or processes, and enables planning, composition and agent/service interoperation; and the *process control model*, which allows agents to monitor the execution of a service request. The first part is referred to as the Process Ontology and the second as the Process Control Ontology. Only the former has been defined in the current version of DAML-S.

It is clear that DAML-S defines an upper ontology for services and it is not complete yet. Also DAML-S service-grounding specification is overlapping with a well-established standard, namely the Web Service Description Language (WSDL) [40].

On the other hand there are certain functionalities that we believe the Web services should provide. The first one is describing a standard way of relating the Web services with electronic catalogs. Secondly, we believe it is necessary to discover services with complementary functionality. For example a "delivery" service may be complementary to a "sell" service, and it should be possible to discover the related complementary ``delivery" service instances given a "sell" service instance. We project that more research and development work is needed for Web service descriptions.

There are two more issues to be handled once the semantics is defined: First, there should be a mechanism in the service registry to relate the semantics defined with service advertised. Secondly, there should be a query facility or better yet, standardized queries to retrieve the necessary information from the ontology definition. We address these issues in Section 4.

3. AGENT TECHNOLOGY

Once the Web service semantics are available, the software agents can automate service discovery, execution, service composition and monitoring according to user's context. There are numerous agent development platforms that can be used as the infrastructure of the proposed framework complying with well-established standards, like FIPA [10]. We intend to use such a FIPA compliant agent infrastructure, namely JADE [14] as our agent development platform. JADE is multi-agent development framework realized by Telecom Italia Lab. The main aim of JADE is to simplify development of agent-based systems and ensuring FIPA standards compliance through a set of system services and agents. This goal is achieved by offering the following features to the agent programmer:

- FIPA-compliant agent platform including the main components of FIPA, namely, Agent Management System (AMS), Directory Facilitator (DF), Agent Communication Channel (ACC). These three agents are by default inserted into the run-time system to manage multi-agent system coordination, agent communication, discovery and agent naming service.
- JADE offers a distributed agent platform. That is, the platform can be split on several hosts. At each host only one Java Virtual Machine (JVM) is executed and agents are created as threads on each host to increase system performance. Java events are used for agent-to-agent communication on the same host and Java RMI across hosts. An agent may divide its tasks for parallel execution and JADE schedules these tasks more efficiently than JVM does for threads.
- More than one FIPA-compliant Directory Facilitators can be started in order to implement multi-domain applications logically divided into domains.
- FIPA-compliant IIOP (Internet Inter Object Request Broker Protocol) to connect different FIPA-compliant agent platforms.
- Automatic conversion of messages to/from FIPA-compliant string format from/to encoded transferable encoded Java objects.
- A library of FIPA agent interaction protocols ready to be used by agents.
- Agent registration and naming service through automatic GUID (Globally Unique Identifier) facility.

For mobile devices, we intend to use LEAP [17], which addresses the need for open infrastructures and services to support dynamic, mobile enterprises. LEAP is developing agent-based services supporting three requirements of a mobile enterprise workforce: Knowledge management (anticipating individual knowledge requirements), decentralised work co-ordination (empowering individuals, co-ordinating and trading jobs) and travel management (planning and coordinating individual travel needs). Central to these agent-based services is the need for a standardised Agent Platform. LEAP is developing an agent platform that is: lightweight, executable on small devices such as PDAs and mobile phones; extensible, in size and functionality; and operating system agnostic.

4. SERVICE REGISTRIES

For efficient discovery, Web services need to be advertised in public registries. From this point of view describing the semantics of Web services and service registries like UDDI [32] and ebXML [8] are complementary. However to discover services in the registries through their semantics; there should be a mechanism in the registry to relate the semantics defined with the service advertised. In the following, we first describe two service registries, namely UDDI and ebXML briefly and then address the relationship of the semantics described with services advertised in each of these registries.

Universal Description, Discovery and Integration (UDDI) is jointly proposed by IBM, Microsoft and Ariba. It is a service registry architecture that presents a standard way for businesses to build registries, discover each other, and describe how to interact over the Internet. The UDDI information model, defined through an XML schema, identifies five core types of information. These core types are business, service, binding, service specifications information and relationship information between two parties. Through these data structures, business entities describe information like their name, description, services offered and contact data. Business services provide more detail on each service being offered. Services can then have multiple binding templates, each describing a technical entry point for a service (e.g., mailto, http, ftp, phone, etc.). These structures use category bags for categorization purposes. An item in a category bag contains a tModel key and an associated OverviewDoc element.

More precisely, the use of tModels in UDDI is two-fold:

- Defining the technical fingerprint of services: The primary role that a tModel plays is to represent a technical specification on how to invoke a registered service, providing information on the data being exchanged, the sequence of messages for an operation and the location of the service. Examples of such technical specifications, also termed as service grounding, include WSDL [40] descriptions and RosettaNet PIPs. Note that the tModel mechanism describes only the signature of the services; it does not provide any information on the semantics of the service.
- Providing abstract namespace references: In UDDI, businesses, services and tModels can specify the categories to which they belong in their category bags. Categorization facilitates locating businesses and services by relating them to some well-known industry, product or geographic categorization code sets. Currently UDDI uses the North American Industrial Classification Scheme (NAICS) taxonomy for describing what a business does; the Universal Standard Products and Services Classification (UNSPSC) [35] for describing products and services offered; and ISO 3166, a geographical taxonomy for determining where a business is located. It should be noted that any number of categories could be referenced in category bags.

ebXML, on the other hand, is an initiative from OASIS and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [34]. ebXML registry provides a set of services to manage the repository and to enable information sharing among trading partners. It also allows building classification trees and associating the objects in the registry with these classification schemes via classification objects. Communication with the ebXML registry is done through the ebXML Messaging Service.

In UDDI, tModels provide the ability to describe compliance with a specification, such as a predefined ontology. For example when a particular ontology is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with this ontology. However since there is no way to define relationships between tModels or to query the attributes of tModels (since they do not have any formal description), this is a cumbersome way of relating semantics with services.

On the other hand, ebXML Registry allows building classification trees and association of objects in the registry with these classification schemes via classification objects. In other words ebXML provides necessary mechanisms to support ontologies and therefore is preferable over UDDI registries for describing the semantics of services. However the mechanisms provided by the classification nodes are not enough to support DAML+OIL ontologies and need to be extended.

5. DAML+OIL AND FIPA OKBC

In order to allow agents to talk about and manipulate knowledge, a standard meta-ontology and knowledge model is necessary. FIPA does not enforce any particular language to store and represent the ontology like RDF [RDF, RDFS], KIF [15] or DAML+OIL [6]. It only specifies the language to communicate about ontologies, namely OKBC (Open KnowledgeBase Connectivity) [22]. In this way, the knowledge obtained from or provided to an Ontology Agent is expressed in this knowledge model. The responsibility to translate knowledge from the actual knowledge representation language into and out of OKBC is left to the agents as needed.

The Open Knowledge Base Connectivity provides operations for manipulating knowledge expressed in an implicit representation formalism called the *OKBC Knowledge Model*. The OKBC Knowledge Model supports an object-oriented representation of knowledge and provides a set of representational constructs commonly found in object-oriented knowledge representation systems. OKBC provides

semantic meaning of what a class, slot, facet and frame are, and the operations to query, modify, add or delete classes, slots, facets and frames.

In our platform we specify ontologies through DAML+OIL. However since DAML+OIL provides richer semantic relationships than the OKBC Knowledge Model, there is a need to extend OKBC. For example, DAML+OIL can define two classes to be equivalent. In order to query a DAML+OIL ontology through OKBC to check the equivalency of two classes, OKBC should be enriched with an <equivalentTo,?A,?B> definition whose formal specification in KIF (Knowledge Interchange Format) is provided in the following:

```
(<=> (equivalent-to ?C1 ?C2)
  (and
        (<=> (subclass-of ?C1 ?Cx) (subclass-of ?C2 ?Cx))
        (<=> (superclass-of ?C1 ?Cy) (superclass-of ?C2 ?Cy))
        (<=> (type-of ?C1 ?Ix) (type-of ?C2 ?Ix))
        (<=> (instance-of ?C1 ?Iy) (instance-of ?C2 ?Iy))
  )
)
```

It is necessary to extend OKBC with DAML+OIL extensions by precisely specifying the semantics, in a similar way.

6. PERSONALIZED MOBILE SERVICES AND P3P

For Web services to better serve the needs of the mobile users they need to be personalized according to users' profiles, needs and contexts. This necessitates the users context as well as her profile to be available. Additionally the right of the user to privacy and security must be respected. Furthermore, to relieve the user from burden of specifying her preferences manually, the user profiles should be dynamically created from user's previous click streams and actions. The same user may connect to Web from different devices and therefore if her profile is kept in a client machine, it needs to be transferred each time the user changes his computer. Yet transferring the user profiles between computers may not always be possible due to access rights or availability of the previous computer, let alone locating the device. Furthermore, transferring profiles may not always be possible: different types of mobile devices require data in different formats.

To alleviate these problems we propose a "Trusted Authority" concept [4]. The users register themselves with a trusted authority. The trusted authority in turn keeps the users' profiles as well as their privacy preferences and uses the P3P protocol [23] to preserve their privacy.

P3P provides a simple, automated way for users to gain more control over the disclosure of personal information to the Web services that they use. P3P is designed to help users reach agreements with services. As the first step towards reaching an agreement, a service sends a machine-readable proposal in which the organization responsible for the service declares its identity and privacy practices. The privacy proposal can be automatically parsed by user agents and compared with privacy preferences set by the user. If a proposal matches the user's preferences, the user agent may accept it automatically. If the proposal and preferences are inconsistent, the agent may prompt the user, reject the proposal, send the service an alternative proposal, or ask the service to send another proposal.

7. INTELLIGENT USER INTERFACE AGENT AND SERVICE ORCHESTRATION

To be able to collect user needs, Intelligent User Interface Agents are needed. These agents by accessing user context [28] as well as profiles and by respecting user's rights to privacy and security

determine the necessary Web services to satisfy user needs. They also need to orchestrate the necessary services. The orchestration tool composes the services and monitors their execution. Through this intelligent user interface and the accompanying orchestration tool, the mobile user is relieved from much of the chores in realizing her needs through the Internet.

The systems that use Intelligent User Interface Agents (IUIA) are user adaptive, i.e. such systems attempt to modify their behavior to maximize the productivity of the current users' interaction with the system [11]. Obviously, for any such adaptation to be viable, it must be based upon information gathered concerning the users' current behavior. The problems to be attacked in such systems are: Which aspects of the user's behavior is useful to capture, what information does such behavior give us on the actual intentions and preferences of the user and finally, what use can be made of any such information captured? Within the light of these questions related with the operation of any IUIA some characteristics of these agents may be listed as follows [11]:

- They undertake an information-filtering role based upon perceived user interests.
- They assist the user in communicating their task to the rest of the system. This typically involves presenting the user with an easy to use interface, which hides from them the underlying system's complexities.
- They perform the role of learning the users' profile. Existing Machine Learning techniques might be useful in achieving this task.

8. CONSTRAINTS AND REQUIREMENTS OF MOBILE DEVICES

Mobile devices, in contrast to wired devices, face temporary loss of network connectivity during operation, they discover hosts in an ad-hoc manner; they have limited computing resources such as available battery power, slow processor speed and limited memory, and they are required to dynamically react to network bandwidth changes. One other limiting aspect of most mobile devices is the relative weakness of their human interfaces. Small displays and limited input devices make extensive data entry rather tedious; hence agents that learn and act on behalf of the user become more valuable. To further complicate matters, there are an extensive number of standards related with the mobile network protocols, underlying communication infrastructures, bearers, device operating systems and the markup languages each device type can interpret. For example, there are currently six major cellular network systems/protocols, i.e., Global System for Mobile Communication (GSM), Personal Communications System (PCS with the TDMA and CDMA flavors), General Packet Radio Service (GPRS), Enhanced Data Rates for Global Evolution (EDGE), Universal Mobile Telecommunications System (UMTS) and I-Mode, each having different operational characteristics. In addition, different wireless devices interpret different markup languages, such as Wireless Markup Language (WML), Tiny HTML (tHTML), cHTML, xHTML or HDML. This wide range of technologies provide challenging issues to both the mobile application developers who need to cope with different operating platforms and networking details, and to the organizations which are in need of mobilizing their services to be accessible to the most possible range of mobile devices.

The following constraints of mobile devices has to be taken into consideration in making Web services accessible to mobile devices:

- *Existence of different protocols:* Web services generally run over TCP/IP. However, since there are several bearers and protocols defined for data transmission over a mobile network, mobile services must be easily accessible from these carriers.
- *Resource constraints:* Mobile devices have much lower processor power, less memory, limited display, restricted input devices and a relatively finite energy source when compared to desktops. So, mobile services must be optimized to run on these devices, considering these limitations.

- *Different presentation types:* Besides the number of mobile operating systems, different mobile device types provide support for one or more markup languages or output formats that present data in various ways, depending on the resource capacity of the device. For example, a PDA running Palm OS can present data as Tiny HTML, while a device capable of interpreting Voice XML can allow the to user *listen to* data instead of viewing it. Mobile services must deal with device characteristics and provide support for data formats available on the device on which they are running.
- *Security:* Mobile services need security mechanisms that address user authentication, identity hiding, transaction privacy and data encryption.
- *Principle of operation:* Web services available for mobile devices must handle the fact that mobile devices non-deterministically lose network connectivity much more than wired applications.

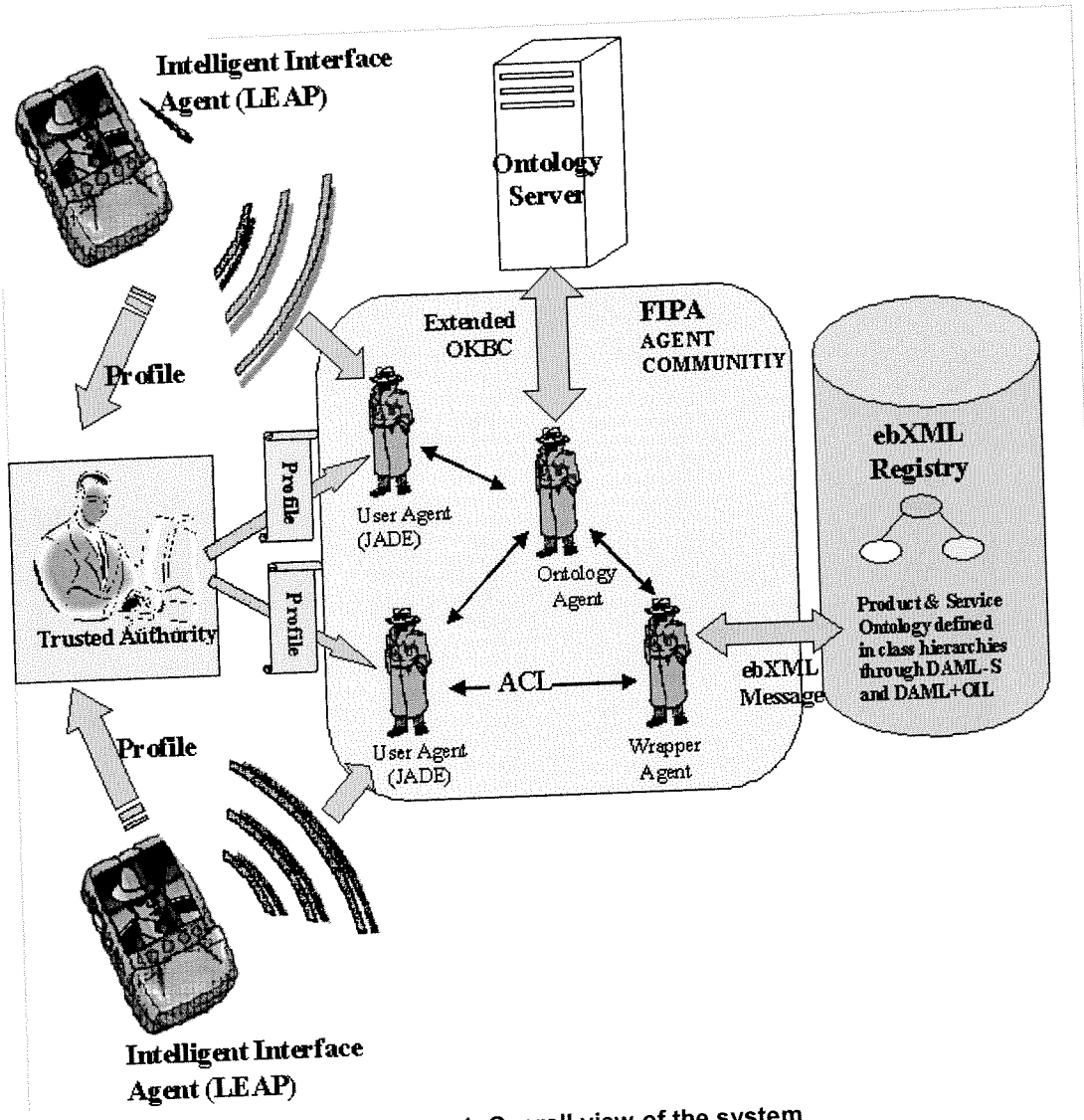


Figure 1. Overall view of the system

9. OVERALL VIEW OF THE SYSTEM

We plan to use the FIPA compliant JADE Agent development environment [14] in our system. However for mobile devices, lightweight agent platforms are necessary and we plan to use LEAP [17](Lightweight Extensible Agent Platform) agents. LEAP descends from JADE and redefines its kernel so that it can run on as small profiles such as phones or watches, while providing the same API as JADE.

Figure 1 demonstrates the overall view of our system. Service descriptions are stored in ebXML registries together with their semantics. Classification nodes of ebXML registries are used to store the Web service ontologies in DAML+OIL. The ebXML registry interface is extended to exploit the semantics of DAML+OIL. Users' profiles are kept in trusted P3P authorities. Each user has a LEAP-based intelligent user agent on their mobile device for collecting user needs and context and then orchestrating the Web services to satisfy those needs by exploiting the semantics of Web services. LEAP agents communicate with JADE agents to convey users' needs, which in turn communicate with trusted authorities to obtain user profiles. Then according to user needs, context and their profiles JADE agents communicate with ebXML registries to compose the necessary services. The system works both in the pull mode and in the push mode.

We also intend to provide DAML+OIL Ontology support to the FIPA Framework. FIPA uses Ontology Agents to share ontologies, which are accessible through OKBC (Open Knowledgebase Connectivity). Therefore to make an ontology conforming to DAML+OIL completely available to a FIPA Ontology Agent, including all the advanced relationships inherent to DAML+OIL, we plan to extend OKBC.

REFERENCES

1. Berners-Lee, T., Hendler, J., Lassila, O. *The Semantic Web*. Scientific American, May 2001.

2. Burg, B. *Agents in the World of Active Web Service*. www.hpl.hp.com/org/stl/maas/docs/HPL-2001-295.pdf

3. *Compact Hyper Text Markup Language Specification*. World Wide Web Consortium, www.w3.org/TR/1998/NOTE-compactHTML-19980209/, February 1999

4. Cingil, I. *Global User Profiles Through Trusted Authorities*. ACM Sigmod Record, Vol.31, No. 1, March 2002.

5. *DARPA Agent Markup Language*. http://www.xml.com/pub/a/2002/01/30/daml1.html

6. *DAML+OIL*. http://www.w3.org/2001/10/daml+oil

7. Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H. DAML Services Coalition. *DAML-S: Semantic Markup for Web Services*. Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.

8. *ebXML, Electronic Business XML*. http://www.ebxml/org/

9. *ebXML Technical Architecture Specification*. http://www.ebxml.org/specs/ ebTA.pdf

10. *FIPA, The Foundation for Intelligent Physical Agents*. http://www.fipa.org/

11. Green, S., Hurst, L., Nangle, B., Cunningham, P., Fergal Somers, Evans, R. *Software Agents: A review*. http://www.cs.tcd.ie/research_groups/aig/iag/pubreview/chap3/chap3. html , 1997

12. *Hyper Text Markup Language 4.01 Specification*. World Wide Web Consortium, http://www.w3.org/TR/html401/, December 1999

13. *Handheld Device Markup Language Specification*. World Wide Web Consortium, http://www.w3.org/TR/hdml20-3.html, March 1997

14. *JADE.* http://sharon.cselt.it/projects/jade/
15. Finin, T., Labrou, Y., Mayfield, J. *A Brief Introduction to Knowledge Interchange Format.* http://www.cs.umbc.edu/kse/kif/kif101.shtml
16. *Knowledge Sharing Effort.* http://www.cs.umbc.edu/kse/
17. *Lightweight Extensible Agent Platform (LEAP).* European Commission's 5th Framework Project, ST-1999-10211, http://leap.crm-paris.com/
18. *OMG Mobile Agent System Interoperability Facility Standard (MASIF).* http://www.fokus.gmd.de/research/cc/ecco/masif/
19. *Ontology Agent.* FIPA Specification XC00086D
20. *Ontology Inference Layer.* http://www.ontoknowledge.org/oil/
21. *Object Management Group.* http://www.omg.org/
22. *Open Knowledge Base Connectivity Specification.* http://www-ksl-svc.stanford.edu:5915/doc/release/okbc/index.html
23. *P3P Platform for Privacy Preferences.* http://www.w3.org/P3P/
24. *Resource Description Framework (RDF) Model and Syntax Specification.* W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax.
25. *Resource Description Framework (RDF) Schema Specification.* W3C Candidate Recommendation, http://www.w3.org/TR/rdf-schema.
26. *The Protege Project.* Stanford Medical Informatics, http://protege.stanford.edu/index.shtml, 2001
27. *RosettaNet.* http://www.rosettanet.org/general/finished-project/laptop.html.
28. Sadeh, N. *A Semantic Web Environment for Context-Aware Mobile Services.* http://www.wireless-world-research.org/BoV1.0/Contributions%2003/03_2_02_Norman_M Sadeh _v2.pdf
29. *Simple Object Access Protocol (SOAP).* http://www.w3.org/TR/SOAP/
30. *SUN ebXML registry.* http://www.sun.com/software/xml/developers/regrep/
31. *SyncML.* http://www.syncml.org/
32. *Universal Description, Discovery and Integration (UDDI).* http://www.uddi.org
33. Universal Mobile Telecommunications System. *Concept Groups for the Definition of the UMTS Terrestial Radio Access concept.* European Telecommunications Standards Institute, TR 101 398 V3.0.1, 1998-10
34. *UN/CEFACT.* http://www.diffuse.org/fora.html\#CEFACT
35. *Universal Standard Products and Services Classification (UNSPSC).* http://eccma.org/unspsc
36. *World Wide Web Consortium.* http://www.w3.org/
37. World Wide Web Consortium. *Requirements for a Web Ontology Language.* http://www.w3.org/TR/2002/WD-webont-req-20020307
38. *Wireless Markup Language Specification.* http://www1.wapforum.org/tech/ documents/WAP-191-WML-20000219-a.pdf, February, 2000
39. *Voice XML Definition.* http://www.oasis-open.org/cover/voxML.html, 2001
40. *Web Service Description Language (WSDL).* http://www.w3.org/TR/wsdl
41. *IBM WSTK Toolkit.* http://alphaworks.ibm.com/tech/webservicestoolkit
42. *Voice XML Specification.* World Wide Web Consortium, http://www.w3.org/TR/2000 /NOTE-voicexml-20000505, May 2000
43. *XHTML.* http://www.xhtml.org/
44. *XML Schema.* http://www.w3.org/XML/Schema
45. *ZEUS Agent Building Toolkit.* http://sourceforge.net/projects/zeusagent

INDUSTRY AND PRACTICE

Issues in Mobile Electronic Commerce

ASUMAN DOGAC, Middle East Technical University
ARIF TUMER, Middle East Technical University

In this article we address delivering highly personalized context sensitive (i.e. time and location dependent) information to mobile clients. We believe providing highly personalized information to mobile clients from a variety of resources like e-mail servers, Web servers, and content providers will be an important service. Mobile network operators can play a major role in delivering this information to users by being strategically positioned between customers and content/service providers. Currently many Mobile Network Operators (MNO) and software companies are offering some personalized services. However the level of personalization is limited to choosing from available content links, icons and services. We believe that the mobile clients will benefit from a much higher level of personalization where it is possible to express complex queries involving variety of data resources. For such a system to gain acceptance by the users, the process of defining personalized information (that is, user profiles) must be very easy for the user. The most important objective of such systems should be to provide scalability, that is, acceptable performance when the number of users increases dramatically. The article also elaborates on the nature of location dependent services.

INTRODUCTION

The inherent limitations of mobile devices necessitate highly personalized information to be delivered to mobile clients. This information may be coming from a variety of resources like Web servers, company intranets, email servers and can be context sensitive (location and time dependent) and need to be delivered to the user according to his profile. Mobile network operators can play a major role in delivering this information to customers by being strategically positioned between customers and content/service providers. Currently many Mobile Network Operators (MNO) and software companies are offering some personalized services based on SMS (Short Message Service) that is available almost all mobile phones. For example Nokia has developed Artus Messaging platform for MNOs, which acts as a gateway between information and applications residing in the Internet or company intranets, and a mobile phone. Messaging platform allows the MNOs to create value-added Wireless Application Protocol (WAP) and messaging applications for all GSM users. For example users are able to select from the available content links and services the operator has provided. This allows each user to personalize and control the information they see on their mobile devices. Other systems available in the market today provide similar services; however the level of personalization is limited to choosing from available content links, icons and services.

We believe that the mobile clients will benefit from a much higher level of personalization. For example, the user may wish to receive an immediate alert if within a period of two hours from the start of the trading day, either IBM stock or Microsoft stock is up in at least 3% more than the change in the Dow Jones index. Such complex requests can only be expressed through query languages and none of the systems on the market today provide this level of personalization. In other words a querying power just like SQL provides on relational databases is necessary for expressing highly personalized profiles. Since the queries will be executed on the documents fetched over the Internet, it is natural to expect the documents to be XML documents, XML being the emerging standard for data exchange over the Internet. Then the user profiles need to be defined through an XML (XML, 1998) query language. XML-QL (Deutsch, Fernandez, Florescu, Levy, & Suciu,1998). is a good candidate in this respect since it has very elaborate mechanisms for specifying query results through the CONSTRUCT statement. This will allow generating the results directly in Wireless Markup Language (WML), ready to be pushed to the mobile clients. A point to be noted here is that the users should not be expected to express their profiles through XML-QL but rather a user-friendly interface should be provided to them to automatically create the XML-QL statements.

When such a system providing highly personalized services is deployed on the Internet, the performance becomes a critical issue since the number of users can easily grow dramatically. A key challenge is then to efficiently and quickly process the potentially huge set user profiles on XML resources. This boils down to developing efficient ways of processing XML-QL queries on XML documents.

The highly personalized profile of the user can also be exploited in delivering location dependent services to mobile clients. For example, when a user is looking for a cinema, the

Manuscript originally submitted

type of pictures that he likes can be obtained from his profile.

In this article we propose an architecture, called MobeCom, to address these issues. The specific aims of the system are as follows:

1. Providing highly personalized information to mobile clients from a variety of resources like e-mail servers, Web servers, and content providers.
2. Providing location dependent services to mobile clients by enriching the directives with online information from the Web like traffic density on the roads. To handle location dependent services a digital map processing system is necessary to locate the user on the map. The data to locate the user on the map can be obtained from a Global Positioning System (GPS), which is available in some of the mobile devices. However if the GPS or a similar mechanism is not available on the user's mobile device, a mechanism should be provided to locate the user on the map by obtaining the street and building number from him. Another important aspect of location dependent services is that it may be necessary to obtain some of this information on line from the Web. For example, stored information about restaurants and cinemas may easily become out of date. Therefore the system should have mechanisms like Internet search agents to obtain information from the Web when needed.
3. For such a system to gain acceptance by the users, the process of defining personalized information (that is, user profiles) must be very easy for the user. For this a set user-friendly graphical interface must be developed both for the user's desktop and also for his mobile device.
4. The most important objective of such systems should be to provide scalability, that is, acceptable performance when the number of users increases dramatically.

RELATED WORK

Recently, push-based, event-driven, content-sensitive information delivery (also termed as **continuous queries** or **push-based data delivery**) has become an active research and development field. Push-based means the active delivery of information without user intervention. Event-driven means that the events of interest that will result in the delivery of data are specified explicitly. Content-sensitive means that information is delivered according to its content (Liu, Pu & Tang,1999).

The execution model of these systems is based on continuously collecting new data items from underlying data sources, filtering them against personalized user profiles (in another terminology evaluating the continuous queries) and finally delivering the result of the queries to the users.

The most notable of examples of push-based data delivery are as follows: OpenCQ project (Liu, Pu & Tang,1999), developed at the Oregon Graduate Institute, employs an SQL-like language and runs on top of a distributed information mediation system which integrates heterogeneous data sources. The system is not scalable since optimization issues have not been considered.

NiagaraCQ system (Chen, DeWitt, Tian & Wang, 2000) developed at the University of Wisconsin, Madison, uses XML-QL and works on distributed XML files. It provides some measure of scalability through query groups, cost based execution plans and cashing techniques. However many issues like what type of index structures specific to XML documents are used or how delta files are obtained and processed are not described. More importantly the performance results are not promising, that is, the execution times of queries are considerably long and this indicates that further research and development is necessary in this area.

A more recent system, Xfilter (Altinel & Franklin, 2000), from UC Berkeley and University of Maryland, uses the Xpath language and works on XML files. This system includes measures for scalability such as index organizations and search algorithms and has impressive performance results. However XFilter, rather than delivering the personalized information, delivers the whole document to the user, which is not feasible in mobile environments.

In (Ozen, Kilic, Altinel & Dogac, 2001), an architecture is provided for mobile network operators to deliver highly personalized information from XML resources to mobile clients. To achieve high scalability in this architecture, the user profiles are indexed rather than the documents because of the excessively large number of profiles expected in the system. In this way all queries that apply to a document at a given time are executed in parallel through a finite state machine (FSM) approach while parsing the document. Furthermore the queries that have the same FSM representation are grouped and only one finite state machine is created for each group. To provide for user friendliness and expressive power, a graphical user interface is developed that translates the user profiles into XML-QL. XML-QL's querying power and its elaborate CONSTRUCT statement allow the format of the results to be specified. The results to be pushed to the mobile clients are converted to Wireless Markup Language (WML) by the delivery component of the system.

Related with location dependent services, there are a number of companies providing real time map support on mobile phones like Webraska (Webraska) and Innovations (Innovations). The challenge in this respect is providing context sensitive services like finding places of interest to the user through the Web by exploiting his or her profile information, locating addresses of these places on the map, providing the user the direction to a destination while processing on line information such as the traffic density on the way and also the opening and closing hours of the places. In (Sharma, 2001) a wide range of wireless Internet applications are described and in (Banerjee, Chrysantis & Pitoura, 2001) data engineering issues for wireless and mobile access are addressed.

A survey of the current state of the mobile commerce is addressed in (Siau, Lim & Shen, 2001). The paper presents an

overview of mobile commerce business activities and technologies, and suggests possible research directions.

SYSTEM ARCHITECTURE

The aim of MobeCom architecture is to deliver highly personalized and context sensitive (time and location dependent) information to mobile clients. Providing highly personalized information to the mobile clients will create better customer satisfaction and can create a multiplier effect and increase the demand for mobile services. Needless to say serving highly personalized information will also to ease the strain on the network bandwidth.

The main components of the system are as follows:

I. Mobile user interface: Mobile users will be presented with a user interface to define their profiles and to receive their location dependent requests, such as the name and the location of the the nearest pizza restaurant or cinema. The interface may get a series of requests from the user with priorities and by proper search on the Web may provide suggestions. The interface will also be capable of giving the user the directions to the destination chosen. The mobile user interface will be developed with Wireless Markup Language (WML).

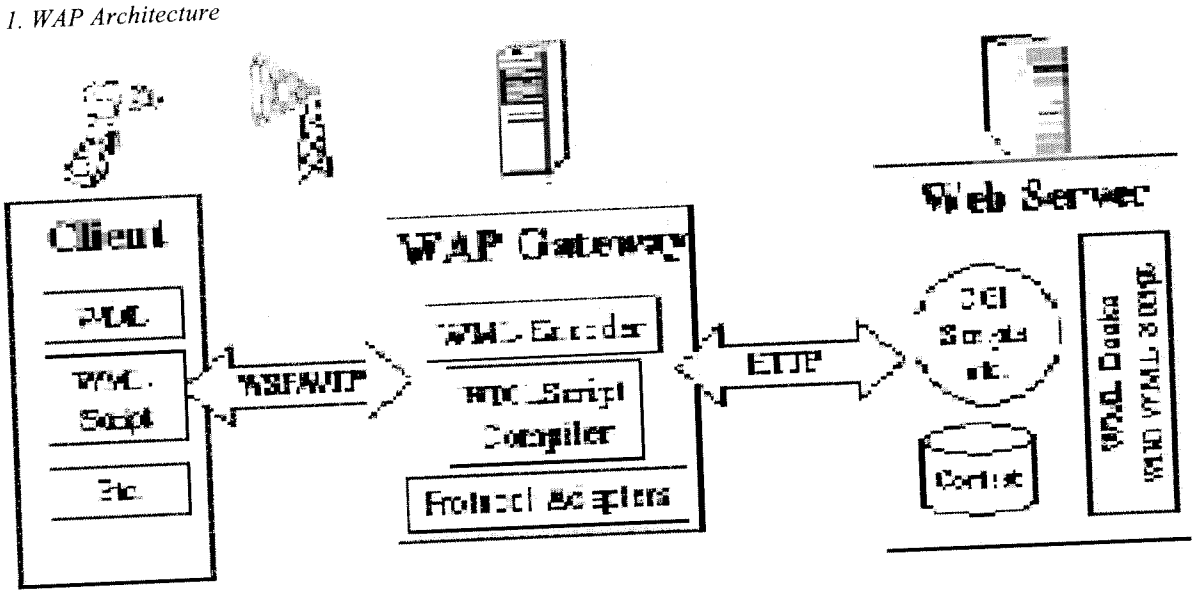
WML is designed for Wireless Application Protocol (WAP) which enables wireless devices to access the Internet directly through a "micro browser". The WAP architecture is shown in Figure 1. Internet standards are inefficient over mobile networks, requiring large amounts of mainly text based data to be sent. Standard HTML web content generally cannot be displayed in an effective way on the small size screens of pocket-sized mobile phone. WAP bridges the gap between the mobile world and the Internet by optimizing Internet standards for the constraints of the wireless environment: low bandwidth, high latency, and less connection stability. WAP enables the delivery of value added services independently of the underlying wireless network technology, bearer and terminal; in this way the mobile terminals together with WAP can access the Internet world. The WML language used for WAP content makes optimum use of small screens and allows easy navigation with one hand without a full keyboard, and has built-in scalability from two-line text displays through to the full graphic screens on smart phones and communicators.

II. Data Collection: Data coming from diverse data resources like channels (news, entertainment, stock prices, etc.), email, Web, file servers will be collected and converted to XML. That is we will provide wrappers for transforming non-XML data to XML. A way of providing XML wrappers for non-XML data is given in (Florescu, Kossmann, Manolescu & Xhuman, 2000). Then XML-QL will be used to extract profile information from the resources.

The data sources could be pull based or push based. Push based data sources will inform MobeCom whenever interesting data is changed. MobeCom will further personalize the data if requested by the user. On the other hand, changes on pull-based data sources need to be checked periodically by MobeCom.

Another interesting issue to be considered is the following: a user may wish to be notified when a composition of events and other situations occur. This issue has been studied in "Amit" middleware framework (Adi, Boltzer, Etzion & Yatzkar-Haham, 2000) which personalizes push technology

Figure 1. WAP Architecture



through event correlation and enables each client to detect customized situations.

III. Personalization: A graphical user interface for personalization will be provided to users both for their mobile devices and their desktops. The profile information provided by the client on the desktops will be mapped to an XML-QL query. XML-QL is one of the candidates being considered by the World Wide Web Consortium (W3) for the standard way of querying XML documents. XML-QL has a SELECT-WHERE construct, like SQL, that can express queries, to extract pieces of data from XML documents. It can also specify transformations that, for example, can map XML data between Document Type Definitions (DTDs) and can integrate XML data from different sources. Although XML-QL shares some functionality with XML's style sheet mechanism, it supports more data-intensive operations, such as joins and aggregates, and has better support for constructing new XML data, which is required by transformations.

Profiles can be classified into two categories depending on the criteria used to trigger their execution. Change based queries are fired as soon as new relevant data becomes available. Timer-based queries are executed only at time intervals specified by the submitting user. The profile specification will be through a user-friendly graphical interface to include the following information (Chen et al, 2000):

CREATE PROFILE name
XML-QL Query
DO action
{START start_time} {EVERY time_interval} {EXPIRE expiration_time}

The query will become effective at the start time. The time_interval indicates how often the query is to be executed. A query is timer-based if its time_interval is not zero; otherwise, it is change-based. Queries will be deleted from the system automatically after their expiration_time. "Action" is performed upon obtaining the query result. For example it could be "Push to +903122105598".

Profile definition on mobile clients will be through Wireless Markup Language. The mobile user interface will be able to get a series of requests from the user with priorities and by proper search on the Web will provide for suggestions for requested items/services. The interface will also be capable of giving the user the directives to the destination chosen.

IV. Providing Scalability: One of the key aspects of this project is providing scalability, that is, the system shows acceptable performance when the number of user's increases dramatically. For this purpose, the following issues need to be addressed:

1. For efficient processing of XML documents, proper index structures should be in place. When indexing XML documents an issue to be decided is whether to index the text file or the DOM the parser creates. Another issue is the type of indices where (McHugh & Widom, 1999) provides some good insights on the type of indices that might be useful. There are also some efforts in indexing XML documents, which may provide further hints (XMLindex).
2. Generating query plans, pruning the search space, grouping similar queries, caching techniques are necessary for effective execution of user profiles on XML documents.
3. Another issue to be tackled is the evaluation of queries on only the changed portions of the updated XML files, which may provide very important performance gains. Some work has been done in this respect at the Stanford University in cooperation with University of Maryland for detecting, representing and querying changes in semi-structured data (Chawathe, Rajaraman, Garcia-Molina, Widom 1996; Chawathe, Abiteboul, Widom, 1999). XML TreeDiff package provided by IBM gives the ability to efficiently differentiate and update DOM trees (XMLtreediff).
4. The work presented in (Ozen, Kilic, Altinel & Dogac, 2001), on the other hand, seems very promising for providing scalability. In this paper, the queries are grouped and indexed such that each element in a query group corresponds to a state in the Finite State Machine (FSM). The system also contains an XML repository. When either there is a change in related XML documents or a timer based query (or a set of queries) needs to be invoked, an event based XML parser is activated that starts sending the events to the Query Execution Engine component of the system that causes the related FSMs to change their states. The Query Execution Engine is capable of capturing the intermediate results during state changes. It should be noted that all the queries that apply to a document are executed in parallel when a document is being parsed and for queries that have the same FSM representation, only one FSM is generated. The results produced are pushed to the related mobile clients. The system is demonstrated to be highly scalable with excellent performance. However there seems to be a need for further investigating executing queries only on the changed data (delta files) rather than the original data file.

V. Providing Context Sensitive (Time and Location) Services: A digital map processing system needs to be developed which will provide capabilities like positioning the user and the destination on the map and providing the shortest path to the destination by also considering the information that may effect the travel like the traffic on the roads. There are many commercial GIS products in the market such as ArcView and GeoMedia. These are very professional and powerful products with many capabilities, which are not actually needed for providing location dependent services. Therefore a map processing system need to be developed whose main functionality will be to find a path (route) in a map to the closest place of a sort, relative to the user position (i.e. finding the nearest drug store). We believe that it will be more

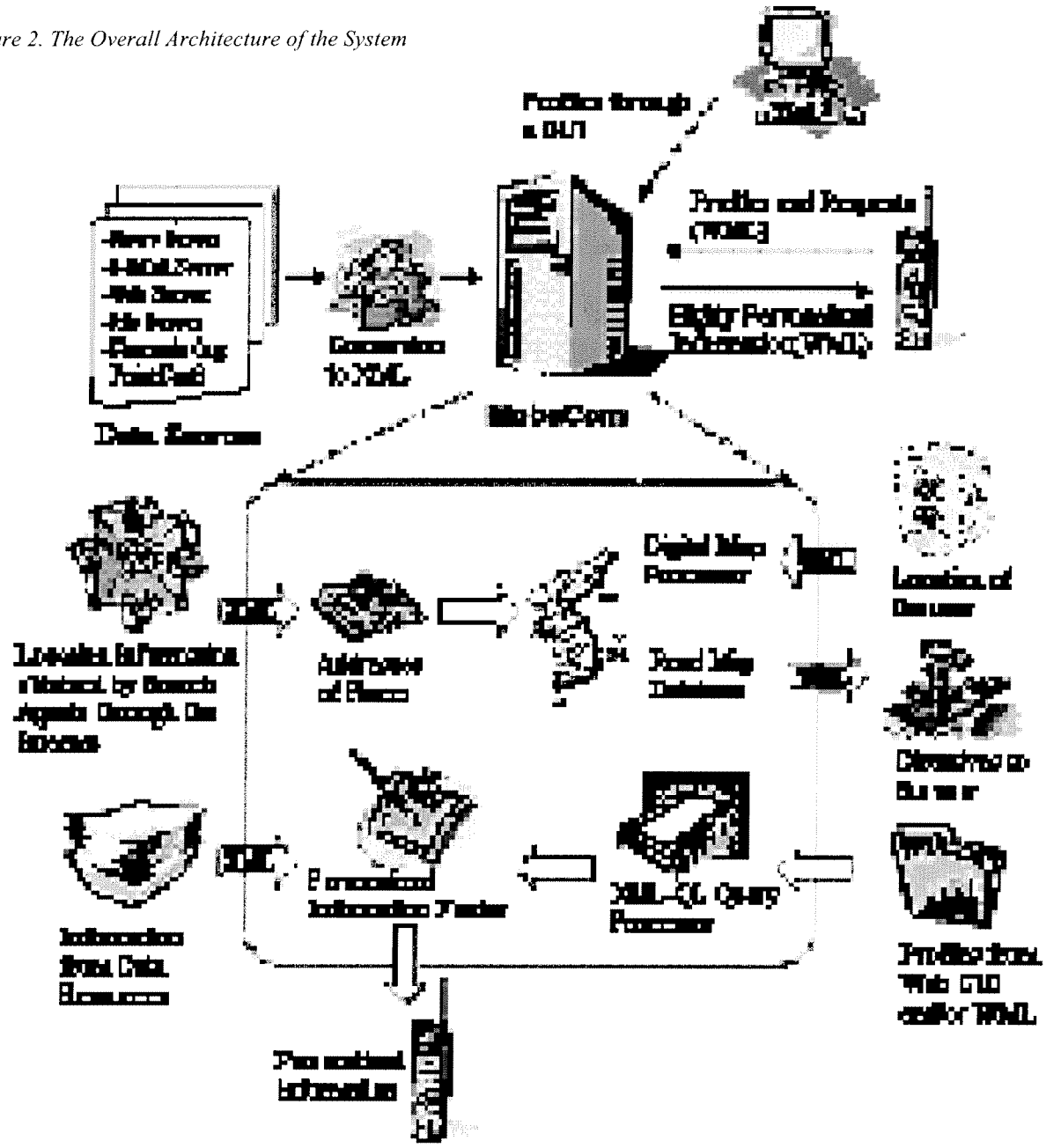
convenient to integrate this custom made Digital Map Processing System with the other modules of the system.

We plan the following digital map processing functionalities:

Describing the map of a part of a city or a whole city in relational database structures: The map of a city (or a region) will be described through relational database structures. A digital road map is basically composed of *arcs* and *nodes*. The *nodes* are binding and bounding points on the map, such as conjunctions or ends of roads. The *arcs* are road segments between the *nodes*. This structure needs to be enhanced with relevant tables in order to store the information like traffic density, road availability, etc.

Finding the shortest path between two points on the

Figure 2. The Overall Architecture of the System



map depending on traffic density and the availability of the roads: One of the shortest path algorithms will be used. The *node* and *arc* tables will include the necessary attributes in order to reflect the traffic information on the map. For bookkeeping the traffic information, heuristics will be used, like assigning a *foreseen* traffic density on a road for each time interval in a day.

Mapping a set of addresses obtained from the Web to a map and finding a path among these nodes that can be travelled with the least effort (i.e. shortest time). The addresses of the places will be mapped by creating virtual nodes on the digital road map. For this purpose each arc segment will have attributes representing the intervals of the numbers of the buildings they contain, i.e. the street numbers of the buildings at the beginning of an *arc* and at the end of that *arc*. The places will be located with another algorithm based on radial-out search. In this algorithm, starting with the position of the user, the map will be traced for places within an area of a growing circle. The system will try to find the closest place of a certain sort, depending on the time that will be spent to reach the destination (the average travel speed on a certain part of a street matters for finding the closest place).

An Internet search engine to gather location dependent information: An Internet search engine will be developed to gather location dependent information on services/products such as finding the service (e.g., a pizza restaurant) the user requested on the Web and locating its address on the map and thus providing the user with a destination.

The overall architecture of the system is depicted in Figure 2. The data obtained (either pulled or pushed by the resources) from the resources are converted to XML and fed into the MobeCom engine. A search engine will also be provided for data requested from unknown resources. On the other hand users specify their profiles either from their desktops or from their mobiles through the user-friendly interface to be provided. Mobile user interface will allow users to specify their location and time dependent requests and can also provide the user with the directives to the desired destination.

For location dependent requests the digital map processing component of the system will be consulted. For positioning the user on the map, a Global Positioning System (GPS) can be used if this is available in user's mobile. If not, a mechanism to locate the user on the map by obtaining the street and building number from him will be in place. The destination(s) satisfying user requests will be obtained from the Web if they are not available in the map database. A map processing system will provide the user the shortest path by also considering the information obtained online from the Web that may effect the travel like the traffic density on the roads, bus schedules, etc.

The user profiles either obtained from the user's mobile device through WML or from the user's desktop through a graphical user interface will be transformed into XML-QL queries. These queries will be executed on the XML documents representing the data coming from a variety of resources. The queries can be change based or timer based. It should be noted that the number of such queries can grow dramatically once the system is deployed over the Internet. Therefore one of the main objectives of the project will be to develop mechanisms like index structures, search algorithms, grouping of queries and caching for efficient execution of these queries.

CONCLUSIONS

Mobile access to Internet is increasing dramatically and therefore serving highly personalized information to mobile clients seems to have a huge market: According to IDC, by 2003, 62 million people will use wireless devices to connect to the Net and the Strategis Group predicts that 25 million users will want cell-phone access to the services like news and sport headlines, stock quotes, email and online shopping. Yahoo and Amazon.com currently offer content to the users of SprintPCS phones through Wireless Application Protocol (WAP). In March 2000, AOL, EarthLink, Microsoft's MSN and Oracle have announced efforts to bring stripped-down versions of their Web portals to the cell phones. In fact Oracle's Portal-to-Go service already lets users access e-mail and check quotes or flight times on the phone's screen. OracleMobile announced that the service they will provide will let the users sign up on their phone as opposed to visiting the Web on a PC first to select what content they want to see on their handset.

What we propose in this article complements these efforts by providing a highly personalized mobile electronic commerce architecture based on XML. We describe the challenges and indicate the future research directions. One big lesson about Web-based electronic commerce is already clear: success comes to those who find ways to deliver to customers orders of magnitude better services. The architecture we propose aims for providing a variety of services for mobile ecommerce and in achieving this, places utmost importance on ease of use and efficiency of the system.

REFERENCES

Adi, A., Boltzer, D., Etzion, O., & Yatzkar-Haham, T. (2000, September). "Push Technology Personalization through Event Correlation". Paper presented at the International Conference on Very Large Data Bases. Cairo, Egypt.

Altinel, M., & Franklin, M. J.(2000, September). "Efficient Filtering of XML Documents for Selective Dissemination of Information". Paper presented at the International Conference on Very Large Data Bases. Cairo, Egypt.

Banerjee, S., Chrysantis, P., Pitoura, E. (2001, May). Proceedings of 2nd ACM Intl. Workshop on Data Engineering for Wireless and Mobile Access. Santa Barbara, USA.

Chen, J., DeWitt, D., Tian, F., & Wang, Y. (2000). "NiagaraCQ: A Scalable Continuous Query System for Internet Databases", Paper presented at the ACM SIGMOD International

Conferance on Data Management, Texas, USA.

Chawathe, S., Rajaraman, A. , Garcia-Molina, H., & Widom, J. (1996, June). "Change detection in hierarchically structured information", Paper presented at the Proc. ACM Sigmod International Conference on Management of Data, pp. 493-504, Montreal, Quebec, Canada.

Chawathe, Abiteboul S., & Widom, J. (1999). Managing Historical Semistructured Data. *Theory and Practice of Object Systems, Vol. 24, No.4.*

Deutsch, A., Fernandez, M., Florescu, D., Levy, A., & Suciu, D,(1998). XML-QL: A query language for XML", W3C Document, http://www.w3.org/ TR/NOTE-xml-ql, 1998.

Florescu, D., Kossmann, D., Manolescu, I., & Xhuman, F. (2000, September). "XML and Relational: How to live with both" Paper presented at the International Conference on Very Large Data Bases. Cairo, Egypt.

Innovations. http://www.innovations.co.uk

Liu, L., Pu, C. & Tang, W. (1999). Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Trans. On Knowledge and Data Enginnering.Vol. 11, No. 4.* 610 - 628

McHugh, J., & Widom, J. (1999, September). "Query Optimization for Semistructured Data", Paper presented at the International Conference on Very Large Data Bases. Edinburgh, Scotland.

Ozen, B. T., Kilic, O., Altinel, M., Dogac, A. (2001, May). "Highly Personalized Information Delivery to Mobile Clients". Paper presented at the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '01). Santa Barbara, USA.

Sharma, C. (2001). Wireless Internet Enterprise Applications. John Wiley & Sons.

Siau, K. , Lim, P. , Shen, Z. (2001,July-September) Mobile Commerce: Promises, Challenges, and Research Agenda, *Journal of Database Management*, 12(3), 4-13.

XML (1998). Extensible Markup Language, http://www.w3.org/XML/, February, 1998.

XMLindex. http://www.xmlindex.com/

XMLtreedif. http://www.alphaworks.ibm.com/tech/xmltreediff.

Webraska. http://www.webraska.com

Asuman Dogac (http://www.srdc.metu.edu.tr/~asuman/) is a professor and the director of Software Research & Development Center at the Computer Engineering Department at the Middle East Technical University, Ankara, Turkey. Her current research interests include electronic commerce, workflow systems and distributed object computing.

Arif Tumer is a graduate student at the Computer Engineering Department of the Middle East Technical University, Ankara, Turkey. His current research interests include mobile electronic commerce.

continued on page 46

Highly Personalized Information Delivery to Mobile Clients

Bahattin Ozen[1] Ozgur Kilic[1] Mehmet Altinel[2] Asuman Dogac[1]
[1]Software Research and Development Center
Middle East Technical University, 06531, Ankara, Turkey

[2]IBM Almaden Research Center, USA

emails: {turhan,ozgur,asuman}@srdc.metu.edu.tr, maltinel@us.ibm.com

Abstract

The inherent limitations of mobile devices necessitate information to be delivered to mobile clients to be highly personalized according to their profiles. This information may be coming from a variety of resources like Web servers, company intranets, email servers. A critical issue for such systems is scalability, that is, the performance of the system should be in acceptable limits when the number of users increases dramatically. Another important issue is being able to express highly personalized information in the user profiles, which requires a querying power as that of SQL on relational databases. Finally, the results should be customized according to user needs, preferences and the mark up language of their mobile device. Since the queries will be executed on the documents fetched over the Internet, it is natural to expect the documents to be XML documents.

This paper describes an architecture for mobile network operators to deliver highly personalized information from XML resources to mobile clients. To achieve high scalability in this architecture, we index the user profiles rather than the documents because of the excessively large number of profiles expected in the system. In this way all queries that apply to a document at a given time are executed in parallel through a finite state machine (FSM) approach while parsing the document. Furthermore the queries that have the same FSM representation are grouped and only one finite state machine is created for each group which contributes to the excellent performance of the system as demonstrated in the performance evaluation section.

To provide for user friendliness and expressive power, we have developed a graphical user interface that translates the user profiles into XML-QL. XML-QL's querying power and its elaborate CONSTRUCT statement allows the format of the results to be specified. The results to be pushed to the mobile clients are converted to the markup language of the mobile device such as Wireless Markup Language (WML), CHTML or XHTML by the delivery component of the system.

1. Introduction

Mobile access to Internet, made possible through Wireless Application Protocol (WAP) [WAP 00], is increasing dramatically with the new GSM extension called GPRS (General Package Radio Service) [GPRS 00] and therefore serving highly personalized information to mobile clients seems to have a huge market: According to IDC [IDC 00], by 2003, 62 million people will use wireless devices to connect to the Internet and the Strategis Group predicts that 25 million users will want cell-phone access to the services like news and sport headlines, stock quotes, email and online shopping.

Mobile network operators play a major role in delivering the information coming from a variety of resources like Web servers, company intranets, email servers to customers by being strategically positioned between customers and content/service providers. The degree of personalization becomes a key issue in such information services due to limited computation power of mobile devices and overwhelming number of potential users. Currently many Mobile Network Operators (MNO) and software companies are offering some personalized services based on SMS (Short Message Service) that is available in almost all mobile phones. For example Nokia has developed Artus Messaging platform for MNOs, which acts as a gateway between information and applications residing on the Internet or company intranets, and a mobile phone [NA 00]. Messaging platform allows the MNOs to create value-added WAP and messaging applications for all mobile users where users are able to select from the available content links and services the Operator has provided. This allows each user to personalize and control the information they see on their mobile devices. Other systems available in the market today provide similar services; however the level of personalization is limited to choosing from available content links, icons and services.

We believe that the mobile clients will benefit from a much higher level of personalization. For example, a user may wish to receive an immediate alert if within a period of two hours from the start of the trading day, either IBM stock or Microsoft stock is up in at least 3% more than the change in the Dow Jones index. Such complex requests can only be expressed through query languages and none of the systems on the market today provide this level of personalization. In other words expressing highly personalized profiles need a querying power just like SQL provides on relational databases. Since the queries will be executed on the documents fetched over the Internet, it is natural to expect the documents to be in XML [XML98], XML being the emerging standard for data exchange over the Internet. Then the user profiles need to be defined through an XML query language. XML-QL is a good candidate in this respect due to its expressive power as well as its elaborate mechanisms for specifying query results through the CONSTRUCT statement. A point to be noted here is that the users should not be expected to express their profiles through XML-QL but rather a user-friendly interface should be provided to them to automatically create the XML-QL statements. It should be also noted that the mechanisms presented in this paper are applicable to more recently proposed XML query language XQuery as well. Finally, the results to be pushed to mobile clients need to be converted to the markup language of the device.

When such a system providing highly personalized services is deployed on the Internet, the performance becomes a critical issue since the number of users can easily grow dramatically. A key challenge is then to efficiently and quickly process the potentially huge set of user profiles on XML resources. This boils down to developing efficient ways of processing large number of XML-QL queries on XML documents.

Although querying XML documents has been a very active research and development issue recently such as in [FK99, FKMX00, CDTW00], there is no consensus on where to store the XML documents (files, relational databases, object-oriented databases), what should be the corresponding schema and the index structures and how to optimize the queries.

On the other hand, the problem at hand is different from these approaches in the sense that to provide for scalability in such an architecture where the critical issue is the very large number of queries, it makes sense to index the queries rather than documents.

In the work described in this paper, which is being performed as a part of CQMC (Continuous Queries for Mobile Clients) project, such an approach is taken. The users are provided graphical user interfaces to define their profiles from their desktops. Simple profiles can also be defined from mobile devices. These profiles are converted into XML-QL queries. The queries can be change based or timer queries; that is, they need to be activated either when the related XML documents change or the time to execute the query expires. The queries are grouped and indexed such that each element in a query group corresponds to a state in the Finite State Machine (FSM). The system also contains an XML repository. When either there is a change in related XML documents or a timer based query (or a set of queries) needs to be invoked, an event based XML parser is activated that starts sending the events to the Query Execution Engine component of the system that causes the related FSMs to change their states. The Query Execution Engine is capable of capturing the intermediate results during state changes. It should be noted that all the queries that apply to a document are executed in parallel when a document is being parsed and for queries that have the same FSM representation, only one FSM is generated. The results produced are pushed to the related mobile clients.

The rest of the paper is organized as follows: Section 2 briefly summarizes the related work. In Section 3, overall architecture of the system is described. The operation of the system, that is how the query index is created, operation of the finite state machine and the generation of the customized results are explained in Section 4. Section 5 gives the performance evaluation of the system. Section 6 investigates how delta files might be used in the architecture proposed. Finally Section 7 concludes the paper.

2. Related Work

The filtering mechanism described in this paper is influenced by the XFilter system [AF 00]. XFilter is designed and implemented for pushing XML documents to users according to their profiles expressed in XML Path Language (XPath) [CD99]. It takes the advantage of embedded schema information in the XML documents to create better user profiles compared to existing keyword based systems. While doing that, it provides efficient filtering of XML documents with the help of profile index structures in its filter engine. XFilter converts each XPath query into a Finite State Machine (FSM) to deal with XPath structures effectively. However as the name implies, Xfilter is a filtering

mechanism; it does not execute the XPath queries to produce partial results. Therefore when a document matches a user's profile, the whole document is pushed to the user. This feature prevents XFilter to be used in mobile environments since the limited capacity of the mobile devices is not enough to handle or process the entire document let alone to receive it.

Furthermore, XFilter does not exploit the commonalities between the queries, i.e. it generates a FSM per query. This observation motivated us to develop mechanisms that uses only a single FSM for the queries which have common element structure. As demonstrated in Section 5, this improvement boosted the system performance drastically. Also the profiles may involve complex queries requiring the use of XML-QL, which has more expressive power compared to XPath. In providing customized results to the mobile clients, the result construction features of XML-QL also help. Finally, timer based queries are an integral part of such systems and this feature is also not available in XFilter.

Another related work is NiagaraCQ system [CDTW00], which uses XML-QL to express user profiles. It provides measures of scalability through query groups and cashing techniques. However, its query grouping capability is based on execution plans which is completely different from our approach and the performance results reported in [CDTW 00], that is, the execution times of queries, do not make such an architecture a possible candidate for mobile environments. Similar to NiagaraCQ, we also replace constants in a query with parameters to be able to create syntactically equivalent queries, which lead to the use of the same FSM for them.

A survey of continuous queries over data streams is presented in [BW 01] which further indicates the research directions in this area.

3. Overall Architecture of the System

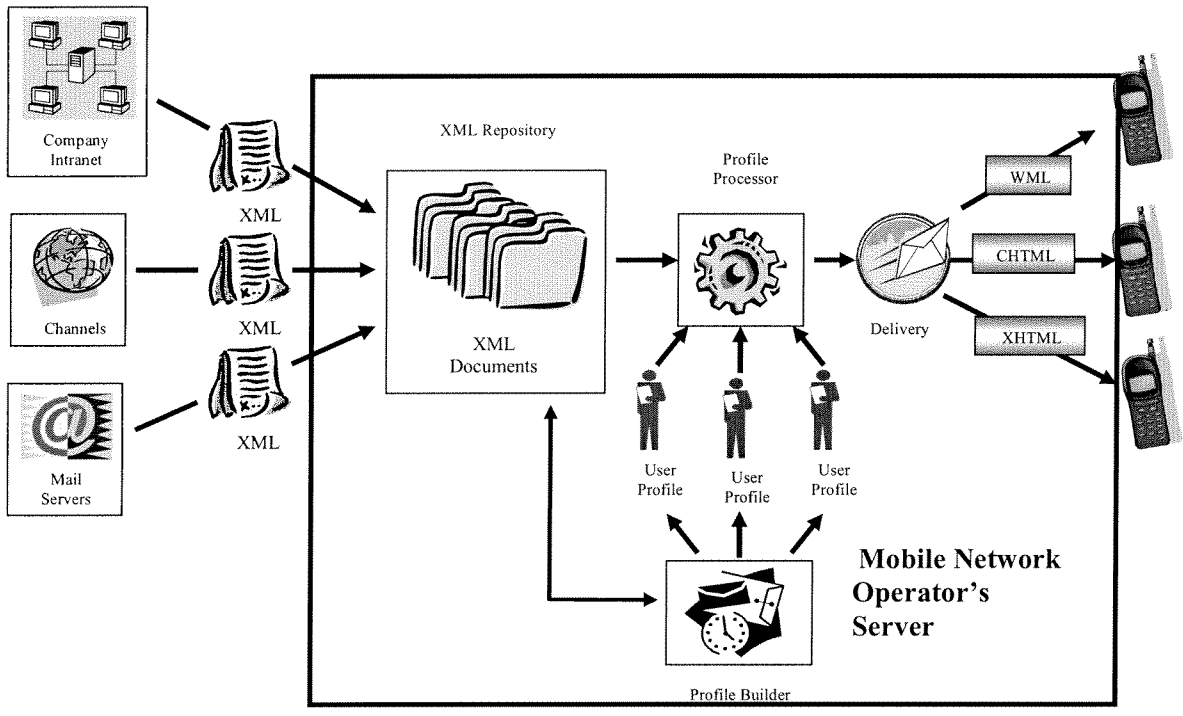


Figure 1. Overall Architecture of the System

The overall architecture of the system is depicted in Figure 1. XML repository contains the XML DTDs and the corresponding data files. Profile Builder component of the system allows for visual query defining capabilities and also manages the user profiles. Profile Processor first creates query indices for user profiles and then parses the documents to obtain the query results. Delivery component of the system pushes the results to the related mobile clients. As indicated in Figure 1, these major system components reside in the server of the Mobile Network Operator (MNO), which is positioned between customers and content/service providers and plays a major role in delivering the information coming from a variety of resources to customers. Among the system components,

Profile Builder is a Web application and hence the browser part of this software resides at the end user device to help with the user to define his profile.

The overall architecture of the system is depicted in Figure 1. XML repository contains the XML DTDs and the corresponding data files. Profile Builder component of the system allows for visual query defining capabilities and also manages the user profiles. Profile Processor first creates query indices for user profiles and then parses the documents to obtain the query results. Delivery component of the system pushes the results to the related mobile clients. As indicated in Figure 1, these major system components reside in the server of the Mobile Network Operator (MNO), which is positioned between customers and content/service providers and plays a major role in delivering the information coming from a variety of resources to customers. Among the system components, Profile Builder is a client/server application and hence a part of this software resides at the mobile device to help with the user to define his profile.

The main components of the system are explained briefly in the following:

a. XML Repository: XML data coming from diverse data resources like channels (news, entertainment, stock prices, etc.), email, Web, file servers are stored as files in an XML repository. The main functionality of XML repository manager is to inform the Profile Processor when it receives a new XML DTD or an XML file.

The data sources could be pull based or push based. Push based data sources inform the repository whenever interesting data is changed. On the other hand the repository manager checks changes on pull-based data sources periodically.

b. Profile Builder: Profile Builder provides facilities through which a user can see his current queries in a list and manage them by adding new queries, activating/deactivating a query defining his profile.

The first phase of a profile creation process is resource selection. Users select the resource XML documents, stored in the repository, on which they want their queries to be executed. In the resource selection part, the DTDs available in the system and XML files confirming to these DTDs are depicted to the user. The resource selection screen is generated dynamically by parsing an XML document that contains the information about available XML documents in the repository grouped by their DTD's. The resource selection screen for an example repository is given in Figure 2. Note that a user may define any number of queries to reflect their profiles.

The resource selection screen lets users to choose the XML documents on which they want their queries to run. The user is able to select a category (i.e. DTD), or a set of XML files. If a category is selected then the XML-QL query is executed on all the XML documents conforming to the selected DTD. The result of the query is the union of the results obtained from each of the XML file(s). The second screen of the Profile Builder, which is dynamically generated according to DTDs, allows a user to create or update a profile based on a selected DTD or selected XML file(s) as shown in Figure 3.

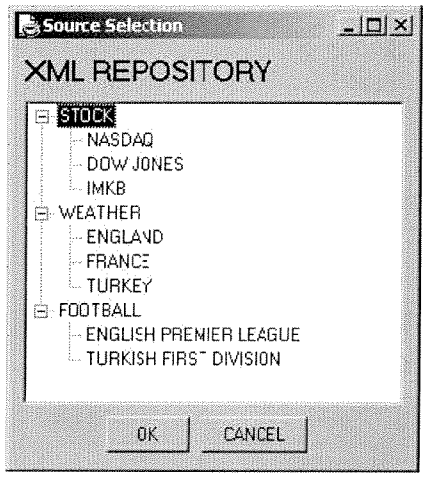


Figure 2. Source Selection Screen

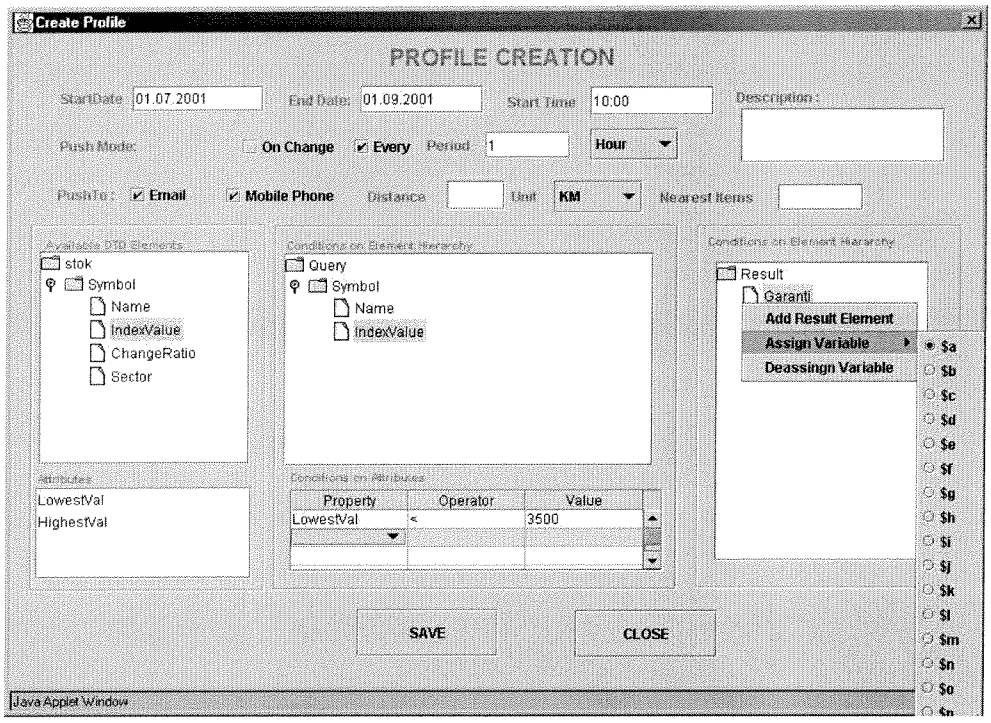


Figure 3. Visual Profile Builder

Queries in a user profile can be classified into two categories depending on the criteria used to trigger their execution. Change based queries are fired as soon as new relevant data becomes available. Timer-based queries are executed only at time intervals specified by the user. The query becomes effective at the start time. The time interval indicates how often the query is to be executed. Queries are deleted from the system automatically after their expiration time. "Push mode" specifies both execution mode of the query and where to send the query result which could be an email address or a mobile phone number. Profiles defined through Visual Profile Builder are transformed into XML documents which contain XML-QL queries as shown in Figure 4.

```
<Profile>
  <XML-QL>
              WHERE  <Symbol> <Name >GARAN</Name>
                              <IndexValue LowestVal=$b >$a</IndexValue>
                     </Symbol>
                     $b < 3500
              IN "imkb.xml"
              CONSTRUCT  <Result>
                                 <Garanti>$a</Garanti>
                         </Result>
  </XML-QL>
  <StartDate> 01.07.2001 </StartDate>
  <StartTime> 10:00    </StartTime>
  <EndDate>  01.09.2001 </EndDate>
  <PushMode onChange="no">
   <Every>  <PeriodSize>1</PeriodSize> <PeriodType>Hour</PeriodType>  </Every>
    <PushTo email="on" mobile="on"/>
  </PushMode>
</Profile>
```

Figure 4. Profile Syntax represented in XML (Grey area shows the XML-QL query)

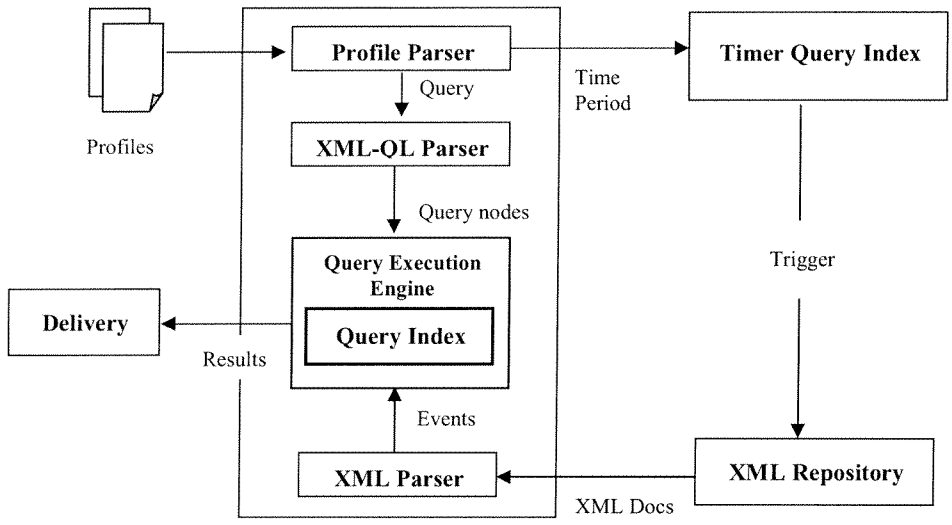


Figure 5. Profile Processor

c. Profile Processor: The basic components of the Profile Processor shown in Figure 5 are as follows: 1) an event-based XML parser, which is implemented using SAX API [Meg98], for XML documents; 2) a profile parser that has an XML-QL parser for user profiles and creates the Query Index; 3) a Query Execution Engine which contains the Query Index which is associated with Finite State Machines to query the XML documents; and 4) a dissemination component that pushes the results to the related users. When a document arrives at the Profile Processor, it is run through an XML Parser that then drives the process of query execution through the query index. The results to be pushed to the mobile clients are converted to the markup language of the mobile device such as CHTML, XHTM, or WML, whereas for the results to be sent to the email addresses a pre-defined style sheet is used.

d. Delivery Component: Delivery component of the system pushes the results to the related users.

4. Operation of the System

The system operates as follows: Profile Builder informs Profile Processor when a new profile is created or updated; the profiles are stored in an XML file that contains XML-QL queries, execution conditions (time-base or change-base), and addresses to dispatch the results (see Figure 4). The Profile Parser component of the Profile Processor parses the profiles; XML-QL queries in the profile are parsed by an XML-QL parser. While parsing the queries, the XML-QL parser creates FSM representation of each query, if the query does not match to any existing query group. Otherwise, the FSM of the corresponding query group is used for the input query. FSM representation contains state nodes for each element name in the queries which are stored in the Query Index.

At the profile parsing time, another data structure called *Timer Query Index* is created which contains a sorted list of time-based queries and a set of pointers to the related XML documents. When the time expires for those queries in the *Timer Query Index*, the Profile Processor is alerted to parse the related XML document. Similarly when a new document arrives, the XML Repository Manager alerts the Profile Processor so that the related XML document is parsed. The event based XML parser sends the events to the Query Execution Engine. The handlers in the Query Execution Engine respond to these events. By moving the FSMs to their next states when current states succeed certain checks like evaluating the attributes, level checking or pattern matching for character data. In the mean time the data in the document that matches the variables are kept in content lists so that when the FSM reaches its final state, all the necessary partial data to produce the results are ready to be formatted and pushed to the related mobile clients. Note that all the queries that are to be executed on an XML document at a given time are executed in parallel in one parsing of the document.

4.1. Creating Query Index

The state changes of a FSM are handled through the two lists associated with each node in the Query Index (See Figure 8): The current nodes of each query are placed on the Candidate List (CL) of the index entry for its corresponding element name. All of the query nodes representing future states are stored in the Wait Lists (WL) of their corresponding element name. Copying a query node from WL to the CL represents a state transition in the FSM. Notice that the node copied to the CL also remains in the WL so that it can be reused by the FSM in future executions of the query since the same element name may reappear in another level in the XML document.

It should be noted that this system is developed to handle very large number of queries and in such a set it is quite probable that there will be queries that have the same tree structure and the same element names, that is, the same FSM representation but different constant values. In this case a single FSM can handle these queries and as demonstrated in Section 5 this greatly enhances the performance of the system.

When the query index is initialized, the first node of each query tree is placed on the CL of the index entry for its respective element name. The remaining elements in the query tree are placed in respective WLs. Query nodes in the CL indicate that the state of the query might change when the XML parser processes the respective elements of these nodes. When the XML parser catches a start element tag and if a node in the CL of the element in the Query Index satisfies level check and attribute check as explained in Section 4.2, and then the nodes of the immediate child elements of this node in the Query Index are copied from WL to CL. The purpose of the level check is to make sure that the element appears in the document at a level that matches the level expected by the query. The attribute check applies any simple expressions that reference the attributes of the element.

Consider an example XML document and its DTD given in Figure 6 and the example queries and their FSM representations given in Figure 7. Note that there is a node in the FSM representation corresponding to each element in the query and the FSM representation's tree structure follows from XML-QL query structure.

```
<!ELEMENT disk (label, contents)>
<!ELEMENT contents (directory*, file*)>
<!ELEMENT directory (name, size, type, contents)>
<!ELEMENT file (name, size, type)>

<disk>
  <label> C </label>
  <contents>
    <directory>
        |          <name> Java Projects </name>  <size> 1 MB </size>
                   <contents>
   Level: 4        <directory> <name> Sources </name>  <size> 5 KB </size>
                       |           <contents>
                                     <file>
                   Level: 6            <name>MySwing.java</name> <size> 5 KB </size> <type>JAVA</type>
                                     </file>
                                   </contents>
                     </directory>
                     <file> <name>Main.exe</name> <size> 5 KB </size> <type>EXE</type> </file>
                   </contents>
      </directory>
      ....
  </contents>
</disk>
```

Figure 6. An Example XML Document and its DTD (disk.xml)

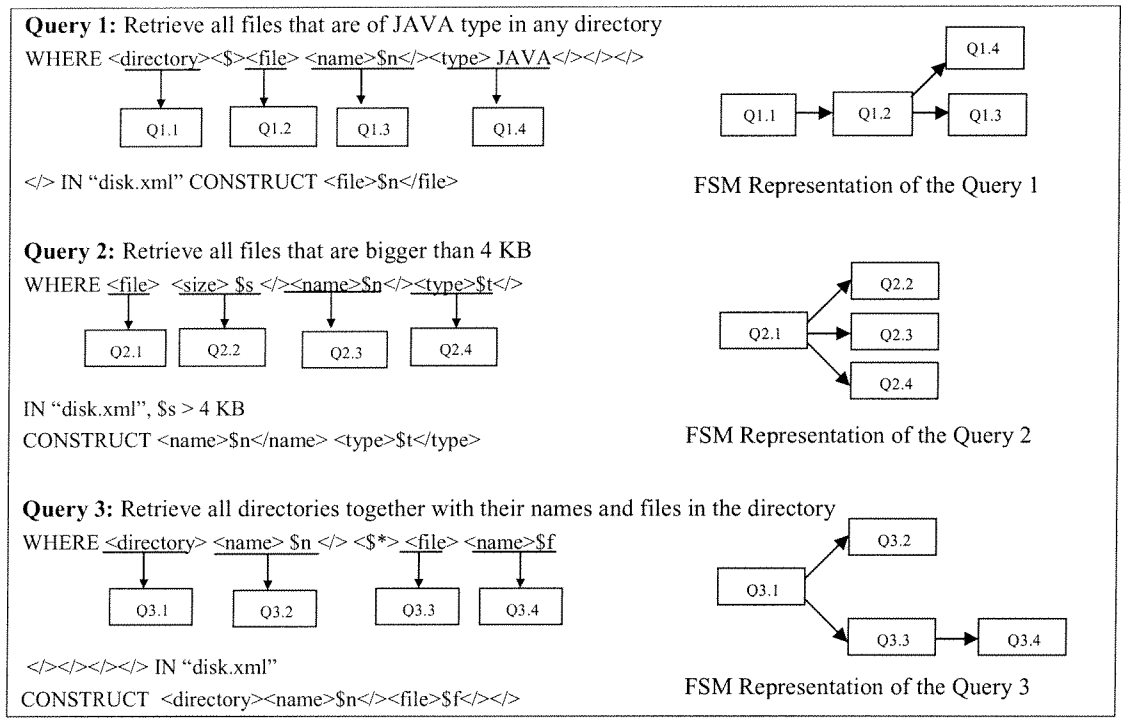
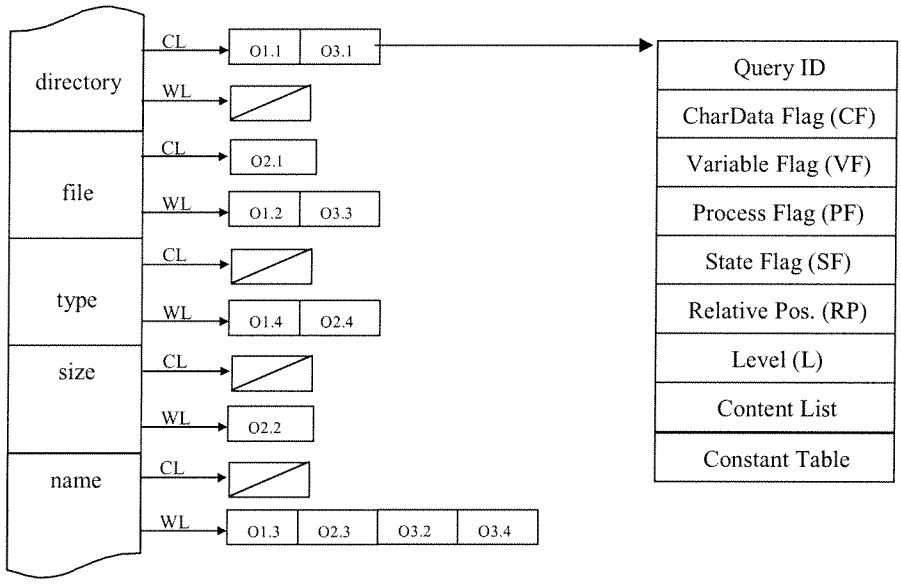


Figure 7. Example Queries

The structure of the query index for the example queries as well as the structure of query nodes is given in Figure 8.



(a) Query Index (b) Node Structure

Figure 8. Initial states of the Query Index for example queries

Each node in each query has a unique identifier. Other elements of the node structure are as follows:

- *CharData Flag* is on when the node in question has a character pattern to be matched,
- *Variable Flag* is on for nodes that have variables,
- *Process Flag* is used for dual purpose; it is set to true for nodes containing character pattern when the start element tag is encountered so that the pattern can be matched in the element data handler. For nodes containing variables, it is set to true when the start element tag is encountered so that the content of the variable can be generated.
- *Level* is an integer that represents the level in the XML document at which this query node should be checked. Because XML does not restrict element types from appearing at multiple levels of a document and because XML-QL allows queries to be specified using "relative" addressing in addition to "absolute" addressing, it is not always possible to assign this value during query parsing. Therefore, this information needs to be updated during the execution of the query.
- *State flag* is set to true when the element of this node is successfully processed that is if the level check, the attribute check and/or data comparison are satisfied.
- *Relative position* is an integer that describes the distance between this query node and the previous (in terms of position) query node in a query tree.
- A *Content List* stores intermediate results for a variable. After finishing XML parsing, outputs are generated from these lists. It should be noted that nested elements can appear in an XML document and therefore generated contents of a variable can occur at different levels. Hence there is a need for another *level* mechanism to distinguish variable contents for elements at different levels. Figure 9 shows the data structure of the content list. A node in the list has a level, content buffer, and a boolean flag called active flag. When start element event is generated, an empty content node is inserted into the content lists of variable nodes and the active flag is set to on. Then, XML data emitted with events is appended into all active contents. In end element event handler, the flag is again set to off position to end content accumulation to the active contents. There is a table in the query execution engine called *Active Content Table*, which keeps tracks of active nodes and provides one shut access while appending data to the active nodes.
- For queries having the same tree structure with same element names but with different constants, only one FSM representation is generated. *Constant Table* is used for such queries to hold the values of different constants as shown in Figure 10.

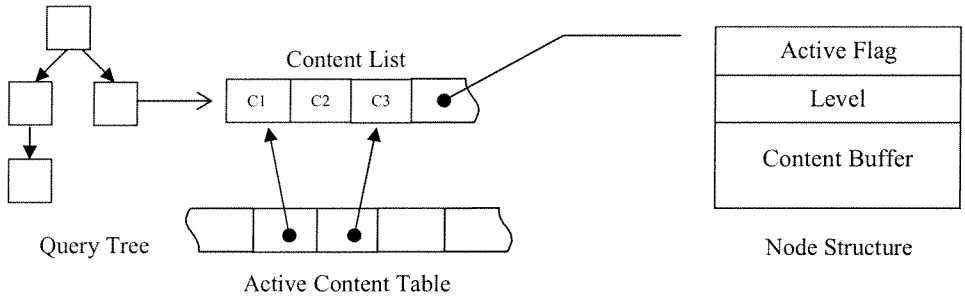


Figure 9. Content List and Its node structure

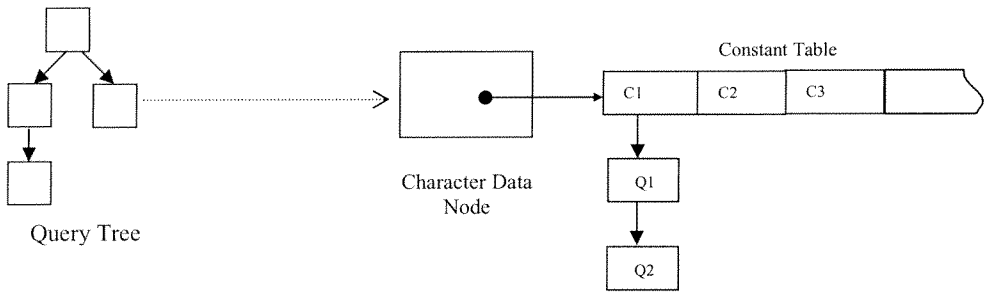


Figure 10. Constant Table

4.2 Operation of the Finite State Machine

When a timer event or a new XML document activates the XML SAX parser, it starts generating events. The following event handlers handle these events:

Start Element Handler checks whether the query element matches the element in the document. For this purpose it performs a level and an attribute check. If these are satisfied, depending on the type of the query node it either enables data comparison or starts variable content generation. As the next step, the nodes in the WL that are the immediate successors of this node are copied to CL at this stage. Before the copy operation, the level value of the new node is calculated with the current level value of the document and relative position value of the current node. Even in a single document, the FSM may be executed more than once if the same element names reappear in the document. Therefore there is a need to reinitialize the FSM. Furthermore XML documents can be nested, that is, the same element may appear at different levels (consider directory in Figure 6). Therefore it may be necessary to generate a FSM to handle this recursion. This is achieved by copying this new node to CL in the query index.

End Element Handler evaluates the state of a node by considering the states of its successor nodes and when the root node is reached it generates the output. End element handler also deletes the nodes from CL which are inserted in the start element handler of the node. This provides "backtracking" in the FSM.

Element Data Handler is implemented for data comparison in the query. If the expression is true, the state of the node is set to true (in state flag) and this value is used by the End Element Handler of the current element node.

End Document Handler signals the end of result generation and passes the results to the Delivery Component.

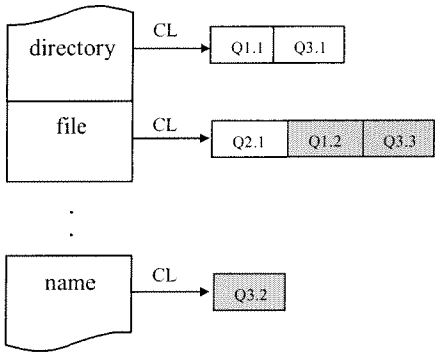
A detailed algorithm of the query execution process is presented in the Appendix I. In this section, we will highlight important steps of the algorithm through an example.

For the document given in Figure 6 and the queries given in the Figure 7 the processing proceeds as follows; the start element tag encountered, *<directory>* at level 4 in the document, causes the state machine for Q1 and Q3 to progress to their next states. Consequently, node Q1.2 and node Q3.3 are copied to CL list of *<file>* element and node Q3.2 is copied to CL list of *<name>* element as shown in Figure 11.a. After the *<directory>* tag, next element in the document is start element tag of *<name>* and Q3.2 is the only node in the CL of *<name>* element. Since Q3.2 contains variable $n, an empty node is inserted into the content list of Q3.2 as shown in Figure 11.b. Then, XML parser reads the character data which happens to be "Java Projects". The Query Execution Engine continues to process the nodes in the CL of *<name>* element. Since Q3.2 is the only node in CL and it contains variable, no pattern matching is done. In this state, character data read is written into the Content Lists of the variable nodes. Afterwards, the level info is updated. Figure 11.c shows the states of the Query Index after processing end element tag *</name>*.

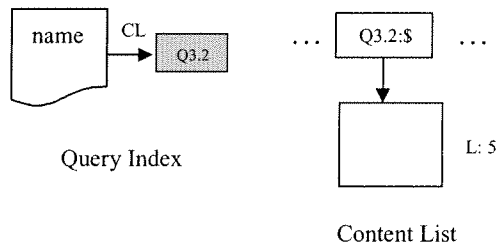
As the parser reports events, the states of queries change as given in the Algorithm Query Execution Process (see Appendix I). When the XML parser sends the event for the start element *<directory>* at level 6 in the document, the Query Index realizes the nested instances of the same element, namely *<directory>*. As given in the algorithm, it puts a copy of Q1.1 and Q3.1 into CL of *<directory>* and updates level info for them. Q1.1° and Q3.1° are the copied nodes shown in Figure 11.d. This copying is necessary to handle multiple nested occurrences of *<directory>*. In this way, more than one FSM are run for a query in the XML document. Also, as given in the algorithm, next nodes of these nodes are copied and are put into corresponding CLs with updated level information.

After processing the start element <*directory*>, at level 6, the current states of Query Index are depicted in Figure 11.d.

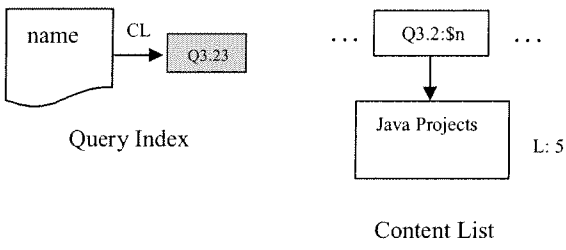
The next two figures, Figure 11 (e) and (f) show CL of <*name*> in the Query Index and content list of Q3.2:$n in the Content list after processing start element tag <*name*> and after processing end element tag </*name*> respectively. Notice that the variable $n of Q3 is partially generated at this step. When the end element tag </*directory*> is encountered and query is satisfied, deactivated content nodes in the content lists of the variables are written into the partial results.



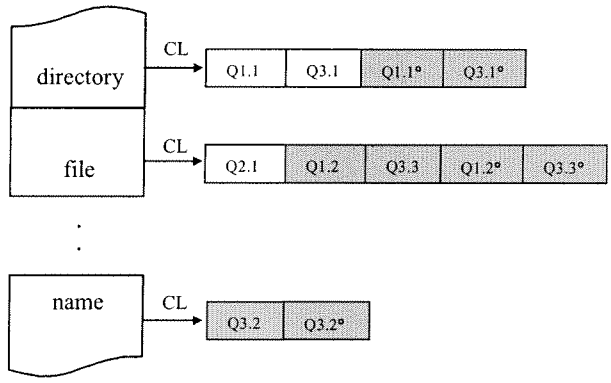
a) After processing <*directory*> at level 4



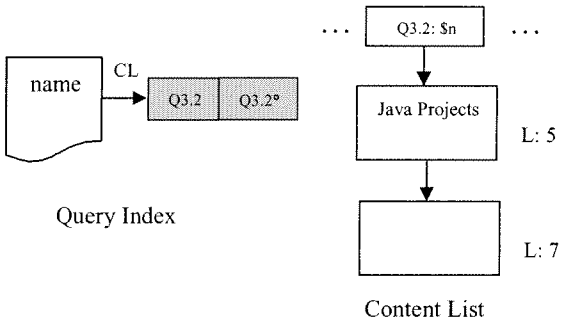
b) After processing <*name*> at level 5



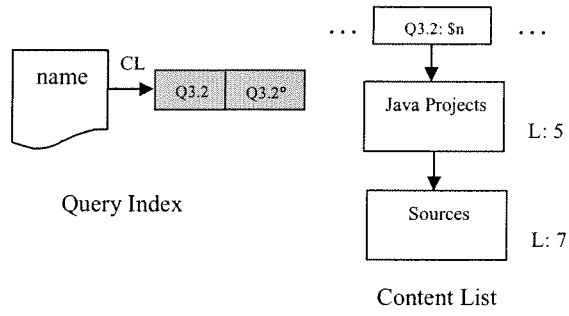
c) After processing </*name*> at level 5



d) After processing <*directory*> at level 6



e) After processing <*name*> at level 7



f) After processing </*name*> at level 7

Figure 11. The states of the Query Index and Content list during the execution

4.3 Generating Customized Results

Results are generated when the end tag element of the query's root element is encountered. Content lists of the variable nodes are traversed to fetch content groups. These content groups are further processed to generate results. This process is repeated until the end of the document is reached.

The results need to be formatted as defined in the CONSTRUCT clause. Therefore while parsing the document to generate the results conforming to the structure defined in the CONSTRUCT clause not only a query tree but also a Construct Tree is generated. An example query and its corresponding Construct tree is depicted in Figure 12.

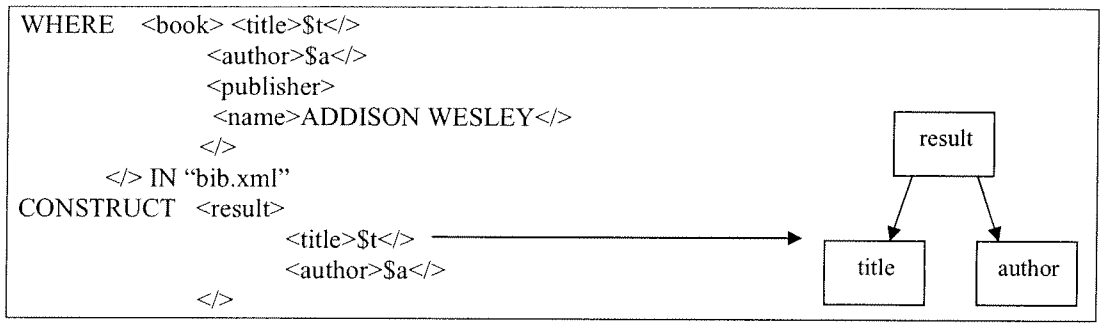


Figure 12. An example query and its corresponding Construct tree

CONSTRUCT part of the query tells how the result sets should be generated and gives the hierarchy of the resulting XML document. We keep this hierarchy in the Construct tree structure. When the end element of the root node is encountered by the parser, it checks whether a result element can be generated by considering the state of the root node of each query. For the satisfying results, construct tree is used for output generation in such a way that collected contents of the variables are merged and the output elements are formed. In Figure 14 this generation is shown for the example query. For each title name in the content list of query node title, an author is selected from the content list of the query node author and a result element is formed.

It should be noted that CONSTRUCT clause in XML-QL may also contain WHERE clauses that query the collected content of the original query, an example of which is given in Figure 13. In this query, the WHERE part of the query when executed produces content for $t and $p variables (see Figure 15). The nested WHERE clause is executed on the content collected for $p to collect content for $a for each $t value. In our architecture this nested query is executed by also forming a query index. Note that there can be several WHERE clauses in the CONSTRUCT clause, each of which recursively uses the collected query content of the former query. When it comes to generating the results, the result of the inner most query is merged with the immediate outer query contents in the Construct tree.

```
WHERE   <book> <title>$t</>
            <publisher><name>ADDISON WESLEY</>
             </>
      </> CONTENT_AS $p IN "bib.xml"
CONSTRUCT  <result>
                  <title>$t</>
                  WHERE  <author>$a</>IN $p
                  CONSTRUCT <author> $a</>
            </>
```

Figure 13. A Sample Nested Query

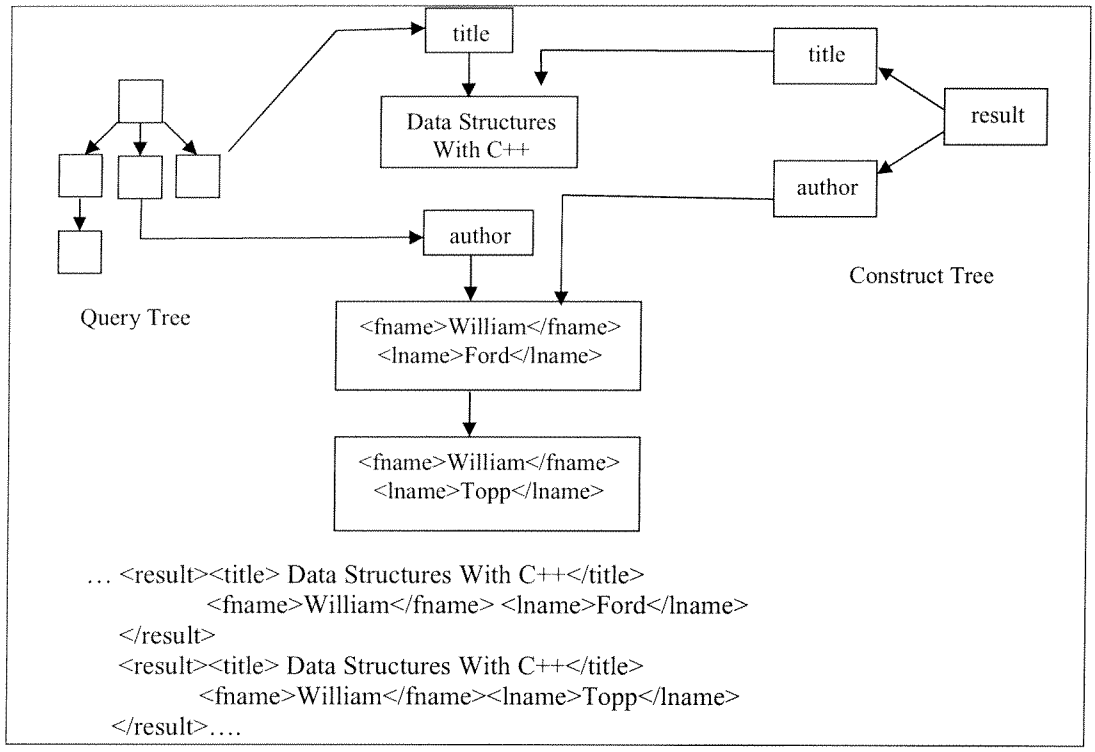


Figure 14. Generated result for the example query

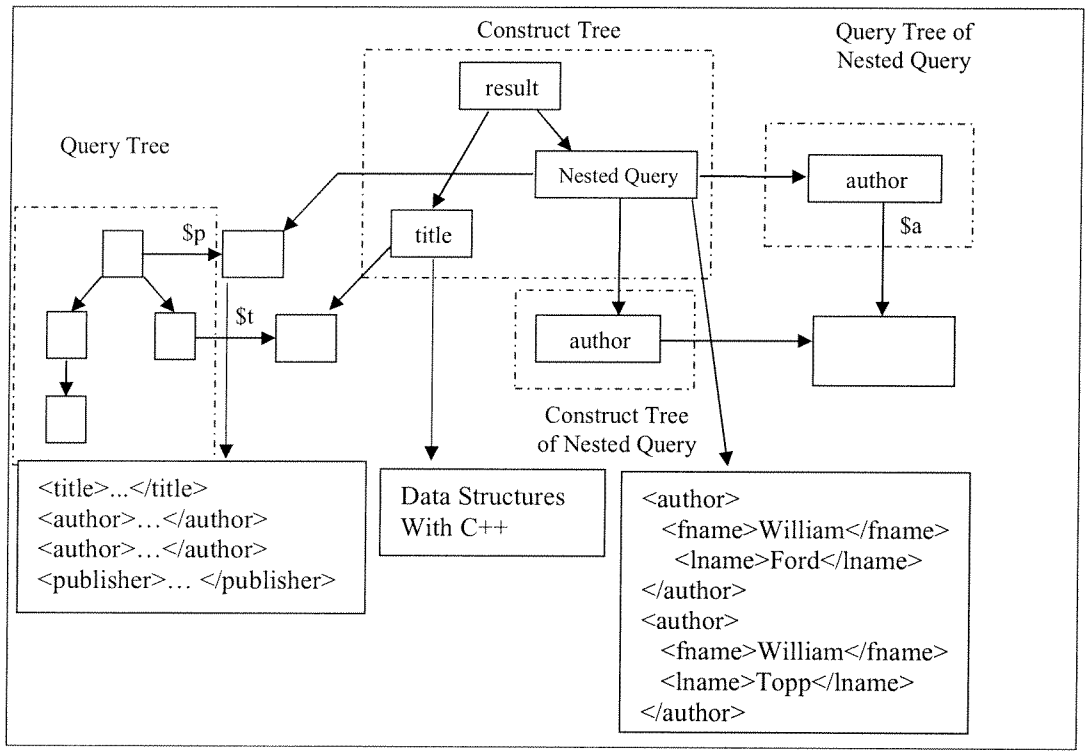


Figure 15. Result Generation for nested queries

4.4 Delivery of the Results

At the end of the XML parsing the result of the each query is written to a file. The Delivery Component gets Ids of queries whose results are ready for delivery operation from the ID Matcher table. For each query whose result is ready for delivery, the result and the XSL [XSL 00] files of the

result document are acquired. ID Matcher table gives only the file names of the profiles. Since the generated result files have the same name with the profile but have a different extension ("rst"), Delivery Component finds corresponding XSL and output filename according to the file name of the profiles. Delivery component also obtains the email address and the mobile phone number of the user from the account information document.

XSL files are used to format the generated result document. These XSL files are constructed during the profile generation step as mentioned in previous sections. Basically, XSL files specify the rules to convert the resulting documents to relevant markup language. Figure 16 shows the delivery component and its interactions.

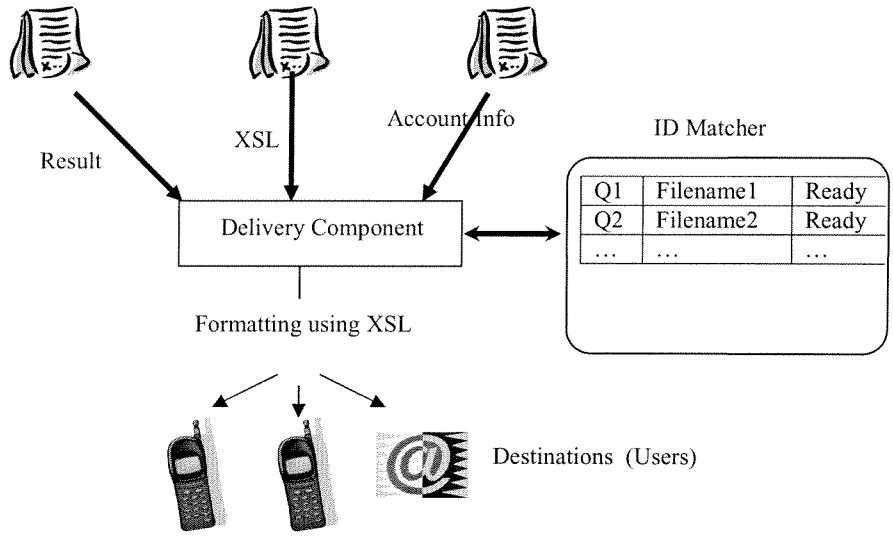


Figure 16. Delivering Results to Users (delivery component and its interactions)

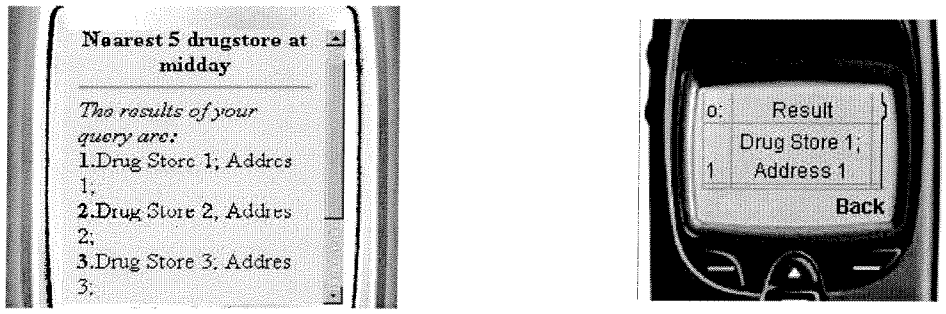
The results of the queries that to be sent to mobile devices are converted into the markup language of the mobile device such as CHTML [CHTML], XHTML [XHTML] and WML [WML 00]. Example style sheets for presenting the result in CHTML and WML are given in Figures 17 and 18. How the results appear in mobile devices is shown in Figure 19. For results that will be sent to e-mail addresses, a pre-defined XSL query is used to generate e-mail messages.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="chtml"/>
<xsl:template match="/">
<html>
        <head>
            <META http-equiv="Content-Type" content="text/html; charset=utf-8">
            <META name="CHTML" content="yes">
            <META name="description" content="result document">
            <title> <xsl:value-of select="/results/description"/> </title>
        </head>
        <body>
        <center><b><xsl:value-of  select="/results/description"/></b></center>
                <hr>
                <I>The results of your query are:</I><br>
                <xsl:for-each select="/results/result">
                        <b><xsl:number value="position()" format="1"/>.</b>
                        <xsl:value-of select="./drugstore/name"/>;
                        <xsl:value-of select="./drugstore/address"/><BR>
                </xsl:for-each>
        </body> </html> </xsl:template> </xsl:stylesheet>
```

Figure 17. Style sheet for CHTML

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="wml"/>
<xsl:template match="/">
<wml>
      <card id="start" title="The results of your query are:">
            <do type="prev" label="Back"><prev />
            </do>            <p>
                  <table columns="2" align="LCC">
                        <tr><td>No:</td><td>Result</td></tr>
                        <xsl:for-each select="/results/result">
                        <tr>
                            <td><xsl:number value="position()" format="1"/></td>
                            <td><xsl:value-of select="./drugstore/name"/>;
                                  <xsl:value-of  select="./drugstore/address"/></td>
                        </tr> </xsl:for-each>
                  </table> </p> </card>
</wml> </xsl:template> </xsl:stylesheet>
```

Figure 18. Style sheet for WML



(a) CHTML (b) WML

Figure 19. Some Presentation Samples for Query Results in Mobile Devices

5. Performance Evaluation of the System

The system is implemented using MS Visual C++ version 6.0. Implementation details can be found in [Kilic 01, Ozen 01]. In this section we present the results of the performance evaluation of the system.

The experiments are conducted on a Pentium III 650 MHz PC with 128 MB memory running MS Windows 2000. All structures are kept in memory (i.e. there is no disk I/O). The characteristics of XML documents used in the tests are given in Table 1. Types of sample queries used in the experiments are given in Figure 20.

Before we start presenting the results of the experiments we find it important to note the following:

1. Our experiment environment (i.e., the Pentium III PC and 128 MB memory) is a modest one. CQMC project is being developed for mobile network operators and obviously, in an industrial environment with a powerful server and a large main memory, the performance of the system will improve.
2. Furthermore the architecture allows to distribute the processing to more than one server very easily.
3. Finally, the performance of the system can further be enhanced by using only the changed data (delta files) rather than the original data file which is investigated in Section 6.

Therefore although the results are presented for a maximum of 100,000 queries (due to memory limitations), we expect that the performance of the system will still be acceptable for mobile environments for millions of queries since the results of the experiments show that the system is highly scalable.

Table 1. Test Data

File	Size	Depth	Number of Elements	Parsing time
Bib1.xml	1 KB	4	35 elements, 7 attributes, 291 characters	1-2 ms
Bib2.xml	50 KB	4	1733 elements, 445 attributes, 15317 characters	40 ms
Bib3.xml	500 KB	4	17475 elements, 4368 attributes, 137141 characters	592 ms
Bib4.xml	1 MB	4	40321 elements, 10080 attributes, 356742 characters	1104 ms

```
WHERE <book> <title>The New New Thing</>
                <author>$a</>
          </>       IN "bib.xml"
CONSTRUCT <author>$a</author>

WHERE <name atr1="1" atr2="2">$a</name>
         IN "bib.xml"
CONSTRUCT <name>$a</name>

WHERE <book><title> $t</>
            <author>
                <firstname>$f</>
                <lastname>$l</>
             </>
        </> CONTENT_AS $p IN "bib.xml"
CONSTRUCT <result><title> $t</>
              WHERE <authors> IN $p
                    <firstname>$f</>
                    <lastname>$l</>
               </authors>

WHERE <book>
            <title>$t</>
             <author>
                <firstname>$f</>
                <lastname>$l</>
             </>
             <publisher><name> Addison-Wesley</></>
        </> IN "bib.xml"
CONSTRUCT <result>
                  <title> $ </>
                   <authors>
                           <firstname>$f</>
                           <lastname>$f</>
                    </>
                </>
```

Figure 20. Sample Queries

We conducted three sets of experiments to demonstrate the performance of the architecture for different document sizes and query workloads.

5.1 Scalability Experiments

The first set of experiments are performed by varying the number of query groups where the test data given Table 1 is used and the number of queries ranged from 50 to 100,000. The graph shown in Figure 21 contains the results for different query groups, that is, the queries have the same FSM representation but different constants, for the document Bib1.xml (1KB). This graph also shows the results of an experiment where the query groups are not taken into consideration. These experiments indicate that proposed architecture is highly scalable and a very important factor on the performance is the number of query groups and that generating a single FSM per query group rather than per query is well justified. It is clear that when you have very large number of queries on the same XML document, many queries will tend to be similar. Consider for example, the users requesting stock quotes; such queries will all have the same FSM representation where only a constant of a condition, namely, the company name will change. And the CQMC system is designed to handle query groups.

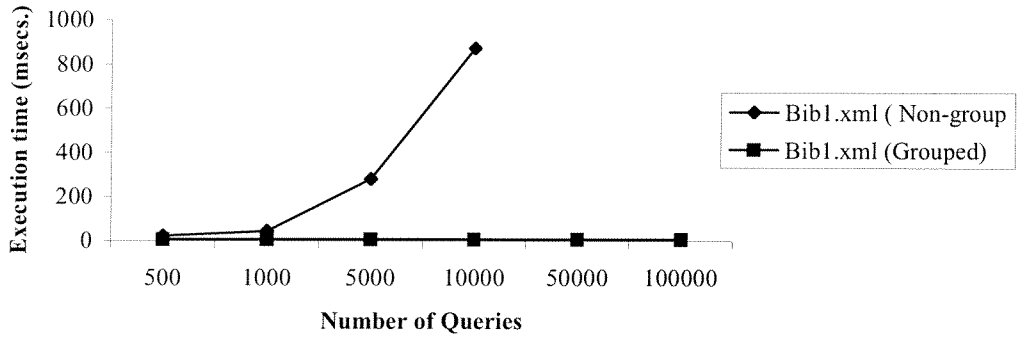


Figure 21. An experiment comparing the performance of grouped queries with non-grouped queries

5.2 Effect of Query Grouping for Different Document Types

In the second set of experiments we fix the number of queries to 100,000 and measure the execution time of the query groups for different size input documents. Figure 22 shows the results for this setting. These results indicate that performance is more sensitive to document size when the number of query groups increases. This result also confirms the importance of the query grouping.

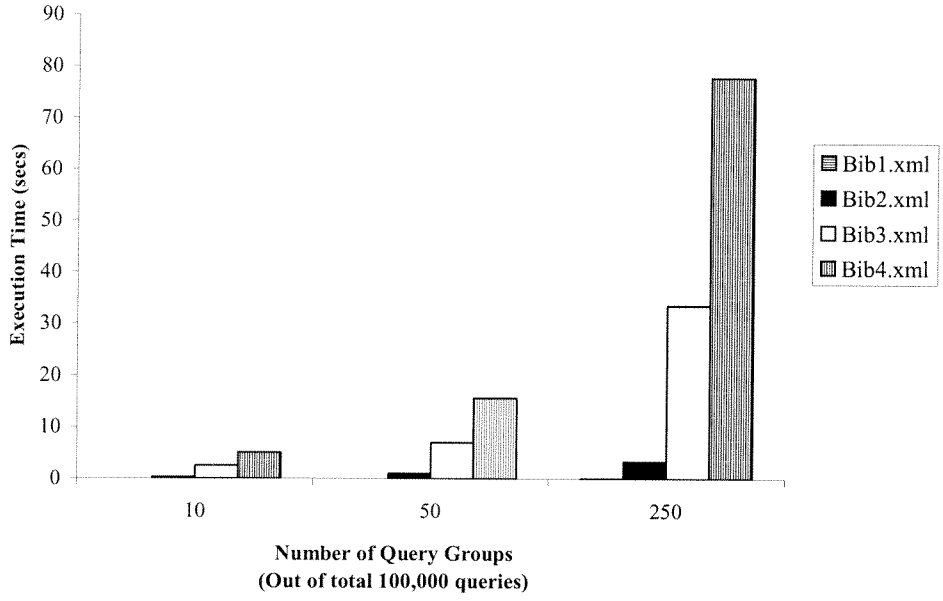


Figure 22. The execution times of queries for different number of query groups and document sizes

5.3 Effect of the Document Size

From the above experiments it is also clear that the size of the document effects the performance considerably. Figure 23 shows how the document size effects the performance. An important observation in these experiments is that large documents degrade the performance drastically when the number query group is high. Therefore, to increase the performance in this respect delta file option needs to be investigated.

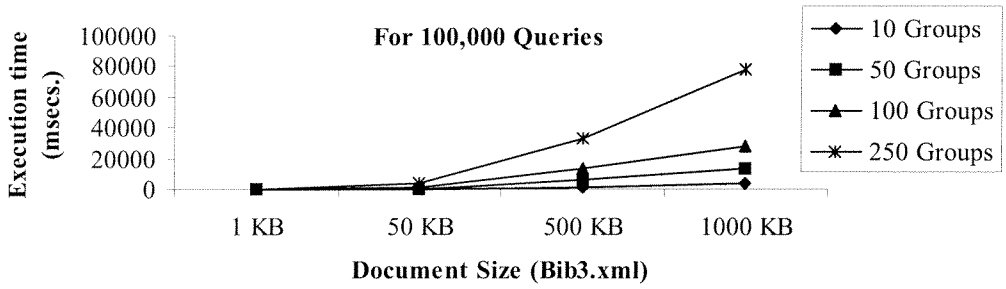


Figure 23. Results showing the effect of document size and query groups on performance

As final conclusion we can say that FSM approach described in this paper for executing XML-QL queries on XML documents residing in memory is a very promising approach to be used in mobile environments.

6. Delta Files

Using delta files rather than the original document is an issue to be investigated. A delta file demonstrates the changes between two versions of a document. A delta file format for XML proposed in [La Fontaine 01] is shown in Figure 24. This format indicates the changes through "modify", "add" or "delete" attributes for each changed element. Note that when there is a change in a child element, this change is propagated through its parents to indicate the change in the child element in the upper levels.

```
<!ELEMENT stock (symbol*)>
<!ELEMENT symbol (indexValue, changeRatio)>
<!ATTLIST symbol
        name CDATA #REQUIRED
        sector CDATA #REQUIRED>
<!ELEMENT indexValue (#PCDATA)>
<!ELEMENT changeRatio (#PCDATA)>
```

Figure 24. The DTD of the Stock.xml

```
<stock d:delta="modify">
        <symbol name="abc" d:delta="modify">
                <indexValue d:delta="modify">
                        <d:PCDATAmodify> <d:PCDATAold>105</d:PCDATAold>
                                         <d:PCDATAnew>110</d:PCDATAnew>
                        </d:PCDATAmodify>
                </ indexValue >
                < changeRatio d:delta="modify">  <d:PCDATAmodify>
                                                    <d:PCDATAold>0.05</d:PCDATAold>
                                                    <d:PCDATAnew>0.1</d:PCDATAnew>
                        </d:PCDATAmodify>
                </ changeRatio >
        </ symbol >...  </stock>
```

Figure 25. An example delta file

The Query Execution Engine of the CQMC system requires minor modifications to handle the additional tags (delta tags) to process delta files. The algorithm handling the delta files is presented in the Appendix II. In the following we demonstrate the process through an example. Consider the query given in Figure 26 and the delta file given in Figure 25 corresponding to the DTD given in Figure 24.

```
WHERE
<symbol name ="abc">
        <indexValue>$a</indexValue>
        < changeRatio>$b</ changeRatio>
</symbol>
CONSTRUCT
<result>
        <abc>
        <indexValue>$a</indexValue>
        < changeRatio>$b</ changeRatio>
        </abc>
</result>
```

Figure 26. An example query

Note that since both the old and the new values of elements satisfy the query condition which is name= "abc", the result set obtained from the new data in delta file can be used to replace the corresponding elements in the original result set.

However it should be noted that not all queries can be handled through delta files. Consider the example query in Figure 27. Since the delta file shows only the changed portions of the XML file, and the "sector" attribute of the element "symbol" has not changed in the document, there is no way to run the query given in Figure 26 on the delta file although its result might have been effected.

```
WHERE
<symbol sector ="xyz">
</symbol> ELEMENT AS $a
CONSTRUCT
<result>$a </result>
```

Figure 27. An example query

There seems to be an obvious advantage to use a delta file rather than the original document when the size of the delta file is a lot smaller than the original file. However, as demonstrated above, whether a delta file will contribute to the result is not straightforward. Deciding on whether to use delta file involves a performance tradeoff depending on several metrics such as the size of the original file, the size of delta file, the number of queries involved as well as the complexity of the algorithm to check the applicability of the delta file to the queries at hand which are the issues that need to be further investigated.

7. Summary and Conclusions

Mobile communication is booming and access to Internet from mobile devices has become possible. Given this new technology, researchers and developers are in the process of figuring out what users really want to do anytime from anywhere and determining how to make this possible.

We anticipate that one of the common uses of mobile devices will be to deliver highly personalized information. There are in fact extensive efforts in this direction, however the level of personalization is limited to choosing from available content links, icon and services. We believe that a querying power is necessary for expressing highly personalized user profiles and for the system to be of use to millions of mobile users, it has to be scalable. Since the critical issue is the number of profiles compared to the number of documents, indexing queries rather than documents makes sense.

This paper describes such an architecture for mobile network operators for delivering highly personalized information from XML sources to mobile clients. The users are provided graphical user interfaces to define their profiles from their desktops. Simple profiles can also be defined from

mobile devices through WML. These profiles are converted into XML-QL queries. The queries can be change based or timer queries; that is, they need to be activated either when the related XML documents change or the time to execute the query expires. The queries are grouped and indexed such that each element in a query group corresponds to a state in the Finite State Machine (FSM) corresponding to that query. The system also contains an XML repository. When either there is a change in related XML documents or a timer based query (or a set of queries) expires, an event based XML parser is activated that starts sending the events to the Query Execution Engine component of the system that causes the related FSMs to change their states. The Query Execution Engine captures the intermediate results during state changes. It should be noted that all the queries that apply to a document are executed in parallel when a document is being parsed. The results produced are pushed to the related mobile clients. Experimental results presented in Section 5 proved that the query grouping and indexing mechanisms presented in this paper provide an excellent performance for different query and XML document workloads.

References

[AF 00] M. Altinel, M. J. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information", in Proc. of VLDB 2000, Cairo, September 2000.

[BW 01] S. Banu, J. Widom, "Continuous Queries over Data Streams", ACM Sigmod Record, Vol. 30, No.3, September 2001.

[CD 99] J. Clark, S. DeRose, "XML Path Language (XPath) Version 1.0", W3C Recommendation, http://www.w3.org/TR/xpath, November, 1999.

[CDTW 00] J. Chen, D. DeWitt, F. Tian, Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases", ACM SIGMOD Intl. Conf. on Data Management, Texas, USA, June 2000.

[CHTML] Compact HTML for Small Information Appliances, *http://www.w3.org/TR/1998/Note compactHTML 19980209*, February 1998.

[FK 99] D. Florescu, D. Kossmann, "Storing and Querying XML Data using an RDBMS", IEEE Data Engineering Bulletin, Vol. 22, No.3, pp27-34, 1999.

[FKMX 00] D. Florescu, D. Kossmann, I. Manolescu, F. Xhuman, XML and Relational: How to live with both, in Proc. of VLDB 2000, Cairo, September 2000.

[GPRS 00] http://horizongprs.motorola.com/whatisgprs/whatis1.htm

[IDC 00] http://www.idc.com/eBusiness/press/ EBIZ082200pr.stm

[Kilic 01] O. Kilic, Profile Generation in Continuous Query Environments for Mobile Clients (CQMC), MS Thesis, Dept. of Computer Eng., Middle East Technical University, September 2001.

[La Fontaine 01] Robin La Fontaine, http://www.gca.org/papers/xmleurope2001/papers/pdf/s29-2.pdf, XML Europe 2001 Conference Proceedings, 21-25 May 2001.

[Meg 98] Megginson Technologies, "SAX 1.0: a free API for event-based XML parsing", http://www.megginson.com/SAX/index.html, May, 1998.

[MW 99] J. McHugh, J. Widom, "Query Optimization for Semistructured Data", in Proc. VLDB, 1999.

[NA 00] Nokia Artus Messaging Platform, http://www.nokia.com/networks/17/maxp.html, 2000.

[Ozen 01] T. Ozen,, Profile Processing in Continuous Query Environments for Mobile Clients (CQMC), MS Thesis, Dept. of Computer Eng., Middle East Technical University, September 2001.

[WAP 00] Wireless Application Protocol, http://www.wapforum.org, June, 2000.

[WML 00] Wireless Markup Language Specification, http://www1.wapforum.org/tech/documents/WAP-191-WML-20000219-a.pdf, February, 2000.

[XHTML] XHTML Basic, *http://www.w3.org/TR/2000/REC-xhtml-basic-20001219*, December, 2000.

[XML 98] Extensible Markup Language, http://www.w3.org/XML/, February, 1998.

[XML-QL 98] XML-QL: A Query Language for XML, http://www.w3.org/TR/1998/NOTE-xml-ql-19980819, August, 1998.

[XSL 00] http://www.w3.org/Style/XSL/

Appendix I: Algorithm Query Execution Process

No level check is performed for the root nodes as well nodes having the value –1 for relative position

Start Element Handler

1. Write start element tag into content of variables.
2. Find element in the *Query Index*
   - 2.1 For each node in the *Candidate List* (**CL**) of the element,
     - 2.1.1 Perform Attribute Filter check and Perform level check
       - 2.1.1.1 If the current node is a root node then
         - 2.1.1.1.1 If the node has a level info and current element level > node level, then copy the current node into the **CL.** Make this node the current node (Move cursor to the this node).
         - 2.1.1.1.2 Update level info of the current node
       - 2.1.1.2 If the node has character data, then set Process Flag to **TRUE**
       - 2.1.1.3 If the node has variable, then
         - 2.1.1.3.1 Set Process Flag to **TRUE**
         - 2.1.1.3.2 Create an empty content for the variable and start collecting content.
         - 2.1.1.3.3 If the type of the variable is **ELEMENT_AS,** then write start tag into content of activated variable.
       - 2.1.1.4 Copy the next nodes of the current node into their **CL**s. Update the level of next nodes.

End Element Handler

1. Find element in the Query Index
   - 1.1. For each node (satisfying level check) in the Candidate List (**CL**) of the element
     - 1.1.1. If the node has the variable and Process Flag is **TRUE**
       - 1.1.1.1. If the variable is an **ELEMENT_AS** varibale, then write the end element tag into content of activated variable.
       - 1.1.1.2. Set Process Flag to **FALSE**
       - 1.1.1.3. Stop collecting content
       - 1.1.1.4. If the next nodes of the node are in the **Final State**, or there is no next node
         then state flag of the node is set to **TRUE**
         else state flags of the next nodes are set to **FALSE**
     - 1.1.2 If the node is Normal node (simple tag <a>),
       - 1.1.2.1 If the next nodes of the node are in the **Final State**, or there is no next node
         then State Flag of the node is set to **TRUE**
         else State Flags of the next nodes are set to **FALSE**
     - 1.1.3 If the node is root node, then
       - 1.1.3.1 if the root node is on final state then collected contents in content list is written to output.
       - 1.1.3.2 Reinitialize the nodes of the query (reinitialize FSM)
       - 1.1.3.3 Clean the next nodes of the current node from **CL**
       - 1.1.3.4 If it is the copied root node, then delete all the next nodes and delete the node.
     - 1.1.4 Write end element tag into contents of activated variables

Element Data Handler

1. Write element data into content of activated variables.
2. Find the element in the *Query Index*
   - 2.1 For each node in the **CL**
     - 2.1.1 If it has the character data,
       - 2.1.1.1 If character data is matched then set state flag of the node and state flag of the query group to
         TRUE

End Document Handler

1. Finalize output generation.
3. Signals delivery component.

Appendix II: Algorithm to process delta files

```
for every element in delta file
    if the element is inserted
        if values of the element satisfies the query conditions
            append result obtained from the element to the previous
            result set
    if the element is deleted
        if values of the element satisfies the query conditions
            remove result obtained from the element from the previous
            result set
    if the element is modified
        if old values of the element satisfies the query conditions and
          new values of the element dissatisfies the query conditions
            remove result obtained from the element from the previous
            result set
        if old values of the element dissatisfies the query conditions and
          new values of the element satisfies the query conditions
            append result obtained from the element to the previous
            result set
        if old values of the element satisfies the query conditions and
          new values of the element satisfies the query conditions
            replace result obtained from the element with corresponding one in
        the previous result set
```

A Semantic-Based Web Service Composition Facility for ebXML Registries

Asuman Dogac, Yildiray Kabak, Gokce B. Laleci

Software Research and Development Center, Middle East Technical University, Inonu Bulvari, Campus, 06531, Ankara, Turkey {asuman,yildiray,banu}@srdc.metu.edu.tr[1]

Abstract

Web services have become a main thrust of the IT industry and there has been considerable development in this area. There are two almost universally accepted standards: SOAP (Simple Object Access Protocol) for invoking services and WSDL (Web Services Description Language) for describing the technical specifications of the services. There are well-established service registries like UDDI (Universal Description, Discovery and Integration) by Microsoft and IBM, and ebXML (electronic business XML) by UN/CEFACT. The infrastructures for Web services are also readily available through application servers like IBM's WebSphere, Microsoft's .NET Framework or BEA's Weblogic. All of these application servers provide support for SOAP, WSDL and UDDI connectivity. However to be able to employ the full potential of Web services, their semantic information should be exploited. In this paper we describe a service composition tool which, by using the semantic information stored in ebXML registries, helps to automate the service discovery and composition.

Keywords

ebXML, ebXML classification hierarchies, Web service semantics, Web service composition, WSDL, SOAP

1 Introduction

The future of business-to-business interoperability seems to be in Web services. Through Web services Internet will become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and to provide value-added services. The dynamic enterprise and dynamic value chains become achievable and may be even mandatory for competitive advantage [Bussler, Fensel, Payne, Sycara, 2002].

The well accepted standards like Web Services Description Language (WSDL) [WSDL 2001], Simple Object Access Protocol (SOAP) [SOAP 2000] and service registries, such as UDDI [UDDI 2003] and ebXML [ebXML 2003] make it possible to dynamically invoke Web services. That is, when the service to be used is known, its WSDL description can be accessed from the registry through a program which uses the information in the WSDL description like the interface, binding and operations to dynamically access the service.

However to be able to exploit the Web services to their full potential, their semantic information should be available. An ebXML registry [ebRIM 2002] allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "classification" mechanism. This information can then be used to discover the services by exploiting the ebXML query mechanisms.

In relating the semantics with the services advertised in service registries, there are two key issues: the first one is where to store the generic semantics of the services (which could be a simple taxonomy or a more complex ontology). Since UDDI does not provide an internal mechanism to store generic service semantics, we have chosen ebXML. ebXML, through its

[1] This work is supported by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 102E035

classification hierarchy mechanism allows domain specific ontologies to be stored in the registries. Note that for UDDI registries, domain specific ontologies can be stored by the standard bodies who define them and the server, where the service is defined, can host the semantic description of the service instance.

The second key issue is how to relate the services advertised in the registry with the semantic defined through an ontology. The mechanism to relate semantics with services advertised in the UDDI registries are the tModel and the category bags of registry entries. tModels provide the ability to describe compliance with a taxonomy. The services can put the semantic information in their category bags through the tModels. In ebXML, on the other hand, classification objects explicitly link the services advertised with the nodes of a classification hierarchy.

In this paper, we describe a semantic-based service composition tool which automates service discovery and composition in ebXML registries. Through a graphical user interface provided, the tool allows the ebXML classification hierarchies to be depicted graphically. When a user clicks on a node in the classification hierarchy, the generic properties of the service are revealed to the user. The user can fill in the desired properties of services she is looking for through this GUI. The tool queries the ebXML registry automatically to find the services that satisfy user constraints. Then, through a graphical composition tool, the user is allowed to provide choreography of the selected services.

The paper is organized as follows: Section 2 briefly summarizes the ebXML specification. Section 3 presents the research approach. In Section 4, we describe the design and implementation of the system. Section 5 concludes the paper.

2 ebXML Specification

Electronic Business XML [ebXML 2003] is an initiative from OASIS and United Nations Centre for Trade Facilitation and Electronic Business [UN/CEFACT 2003]. ebXML aims to provide the exchange of electronic business data in Business-to-Business and Business-to-Customer environments. The ebXML specifications provide a framework in which EDI's substantial investments in Business Processes can be preserved in an architecture that exploits XML's technical capabilities. In other words, the initiative leverages from the success of EDI in large businesses, and intends reaching small and medium enterprises. The vision of ebXML is to create a single set of internationally agreed upon technical specification that consists of common XML semantics and related document structures to facilitate global trade.

ebXML builds on the lessons learned from EDI, particularly the need to identify trading partners and messages, and account for all message traffic. ebXML also identifies common data objects, called core components, that allow companies to interchange standard EDI data with XML vocabularies compliant with the ebXML specifications.

The ebXML architecture provides the following functional components:

- Registry/Repository: A registry is a mechanism where business documents and relevant metadata can be registered, and can be retrieved as a result of a query. A registry can be established by an industry group or standards organization. A repository is a location (or a set of distributed locations) where a document pointed at by the registry resides and can be retrieved by conventional means (e.g., http or ftp).
- Trading Partner Information: The Collaboration Protocol Profile (CPP) provides both DTD and XML Schema definitions of an XML document that specifies the details of how an organization is able to conduct business electronically. It specifies such items as how to locate contact and other information about the organization, the types of network and file transport protocols it uses, network addresses, security

implementations, and how it does business (a reference to a Business Process Specification). The Collaboration Protocol Agreement (CPA) specifies the details of how two organizations have agreed to conduct business electronically. It is formed by combining the CPPs of the two organizations.

- Business Process Specification Schema: The Business Process Specification Schema provides the definition of an XML document (in the form of an XML DTD) that describes how an organization conducts its business. While the CPA/CPP deals with the technical aspects of how to conduct business electronically, the Business Process Specification Schema deals with the actual business process.
- Messaging Service: ebXML messaging service provides a standard way to exchange messages between organizations reliably and securely. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP.
- Core Components: ebXML provides a core component architecture where a core component is a general building block that basically can be used to form business documents.

Note that a service in ebXML is represented with a "Service" class in ebXML Registry. The technical specification files (i.e., WSDL descriptions of the service instance) are stored in ebXML registry together with "Service" class as extrinsic objects. The relationship between the description files and the "Service" class is established through the "ServiceBinding" Class of ebXML. A "Service" class may have a collection of "ServiceBinding" classes each of which represents technical information on how to access a specific interface offered by a "Service" instance. Also, a "ServiceBinding" instance has several "SpecificationLink"s each of which provides the linkage between a ServiceBinding and one of its specifications describing how to use the Service.

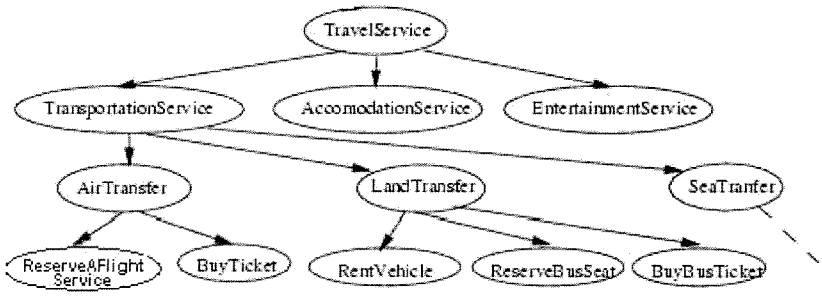


Figure 1 An example classification hierarchy for travel industry

3 The Research Approach

An ebXML compliant registry allows metadata to be stored in the registry. This is achieved through a "classification" mechanism, called *ClassificationScheme* which helps to classify the objects in the registry. *ClassificationScheme* defines a hierarchy of *ClassificationNodes* [ebRIM 2002, ebRSS 2002].

Consider for instance the classification hierarchy example given in Figure 1 for travel industry. When such a hierarchy is stored in an ebXML registry, the registry objects can be related with the nodes in the hierarchy. In this way it is possible to give meaning to the services. In other words, by relating a service with a node in classification hierarchy, we make the service an explicit member of this node and the service inherits the well-defined meaning associated with this node as well as the generic properties defined for this node. As an example, assume that there is a service instance in the ebXML registry, namely, "MyService". When we associate "MyService" with "ReserveAFlightService" node, its meaning becomes clear; that this service is a flight reservation service. Assuming that the

"ReserveAFlightService" service has the generic properties such as "originatingFrom", "destinationTo" and "paymentMethod", "MyService" also inherits these properties.

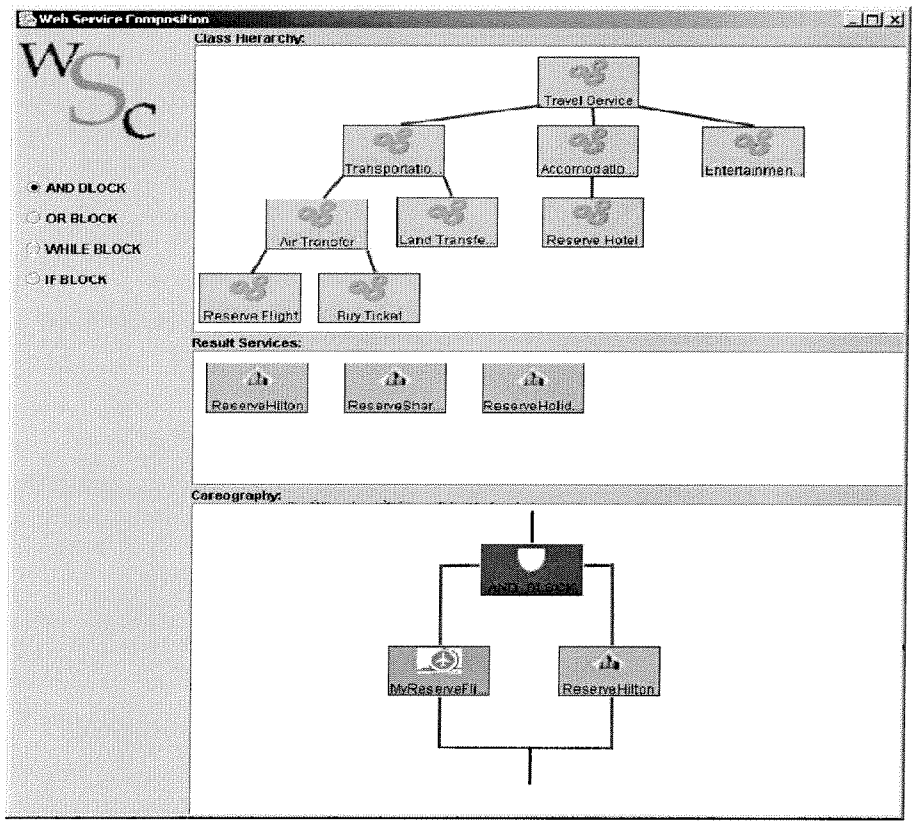


Figure 2 GUI tool for semantic-based Web service composition for ebXML registries

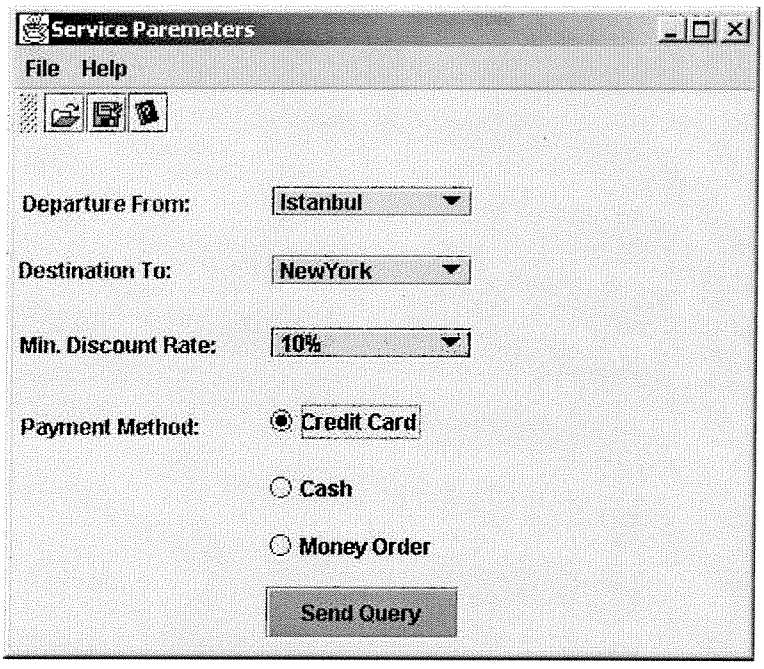


Figure 3 GUI to obtain service property values from the user

Providing precise meaning and properties of the services facilitate their discovery. For example, if we want to find all the flight reservation services in the registry, it is possible to query the services that are classified under the generic "ReserveAFlightService" node.

ebXML query facility can be used for this purpose which supports two query capabilities [ebRSS 2002]: Filter Query and SQL Query. SQL Query support is optional whereas Filter Query is mandatory for a registry to be ebXML compliant. A client submits a Filter Query as part of an "AdhocQueryRequest". We have used Filter Query capability in the implementation.

What an "AdhocQueryRequest" must return is specified with the "ResponseOption". The "ResponseOption" element has an attribute "returnType" whose values indicate the type of the response to be returned such as the identifiers of the registry objects, attributes of the registry objects, or leaf classes. For example, a RegistryObjectQuery with "ResponseOption" "LeafClass" will return all attributes of all instances that satisfy the query.

4 The Design and Implementation

We have developed a tool kit, whose GUI is presented in Figure 2. The top most part of the canvas depicts one of the ebXML classification hierarchies. When a user clicks on a classification node, the slots for this node are depicted to the user as shown in Figure 3 so that the user can provide the preferred values for the slots. Then the system finds the members of this node that satisfy the user given values and the resulting services are listed in the middle of the canvas. The user selects the services among the presented alternatives and forms a workflow by combining the services with the control flow blocks presented.

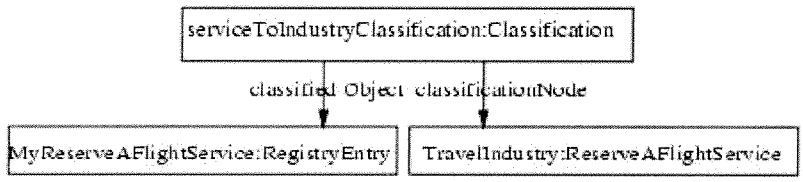


Figure 4 Relating service instances with the nodes in the classification hierarchy

In this section we present the details of this development effort. Each service is published to ebXML Registry via "SubmitObjectRequest". Through "SubmitObjectRequest", it is possible to relate the service instance with a generic node in the classification hierarchy. The WSDL files necessary to invoke the service is also indicated in this request.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SubmitObjectsRequest >
        <rim:LeafRegistryObjectList>
                <rim:ClassificationNode id = 'ReserveAFlightService' parent= 'CS' >
                        <Slot name = 'originatingFrom' slotType= 'StringList'>…</Slot>
                        <Slot name = 'destinationTo' slotType= 'StringList' > …</Slot>
                        <Slot name = 'paymentMethod' slotType= 'StringList' >…</Slot>
                </rim:ClassificationNode>
<Service id="MyReserveAFlightService">
        <Name> <LocalizedString lang="en_US" value = "Reserve Flight Service"/> </Name>
        <Slot name = 'originatingFrom'> <ValueList>
                <Value>Istanbul </Value> </ValueList> </Slot>
        <Slot name = 'destinationTo'> <ValueList>
                <Value>New York</Value> </ValueList> </Slot>
                <Slot name = 'paymentMethod'> <ValueList>
                <Value>Credit Card</Value> </ValueList>    </Slot>
        <ServiceBinding accessURI="http://www.sun.com/ebxmlrr/registry/nameSpaceIndexer">
                <SpecificationLink specificationObject="wsdl">
```

```
                    </SpecificationLink>
            </ServiceBinding> </Service>

<ExtrinsicObject id="wsdl" mimeType="text/xml">     </ExtrinsicObject>
<Classification classificationNode="ReserveAFlightService" classifiedObject="MyReserveAFlightService" />
            </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

Figure 5 "SubmitObjectRequest" which declares the semantic of "MyReserveAFlightService" and relates it with the "ReserveAFlightService"

As an example, assume that there is a service called "MyReserveAFlightService" stored in the ebXML registry and a *ClassificationNode* exist for "ReserveAFlightService". Figure 4 demonstrates how this service is associated with this classification node by using the classification object. In Figure 5 we provide an example "SubmitObjectRequest" which declares the semantic of "MyReserveAFlightService" and relates it with the "ReserveAFlightService" generic node. Note that a registry object can be classified according to any number of classification schemes.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<AdhocQueryRequest xmlns = "urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation = "urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0 query.xsd">
 <ResponseOption returnType = "LeafClass" returnComposedObjects = "true"/>
 <FilterQuery>
  <RegistryObjectQuery>
   <NameBranch>
        <LocalizedStringFilter>
         <Clause>
          <SimpleClause leftArgument = "value">
           <StringClause stringPredicate = "Equal">
                ReserveAFlightService</StringClause>
          </SimpleClause>
         </Clause> </LocalizedStringFilter>
   </NameBranch>  </RegistryObjectQuery>  </FilterQuery> </AdhocQueryRequest>
```

Figure 6 An AdhocQueryRequest to return "ReserveAFlightService" as a composite object including the slots

Once the services are classified with classification nodes in this manner, it is possible to obtain all services that satisfy user given criteria by using ebXML's query facilities. We have developed three query templates for this purpose:

- The first query template retrieves all the properties of a generic node. Given a node in the classification hierarchy, such as "ReserveAFlightService" node, this query, shown in Figure 6, returns the properties of the service class. Note that when the "ResponseOption" is "LeafClass" and "returnComposedObjects" is true, the properties (i.e., the slots) of the object are returned. The tool developed uses these properties to generate a dynamic GUI so that the user can provide her preferred values. An example GUI is provided for the "ReserveAFlightService" is given in Figure 3. Through this GUI, the user fills in her preferred values for the service parameters.
- The second query template is used for locating service instances of a given generic class node in the class hierarchy. Assuming that the user requested all the flight

reservation services originating from Istanbul, the query given in Figure 7 retrieves "ReserveAFlightService" instances "originatingFrom Istanbul".

- The third query template is a content retrieval query to obtain the WSDL files through the identifiers of the WSDL files in the SpecificationLinks.

```
<AdhocQueryRequest >
 <ResponseOption returnType = "LeafClass" returnComposedObjects = "true" />
 <FilterQuery> <ServiceQuery>
  <ClassifiedByBranch>
    <ClassificationNodeQuery> <NameBranch> <LocalizedStringFilter>
       <Clause>
        <SimpleClause leftArgument = "value">
         <StringClause stringPredicate = "Equal">ReserveAFlightService </StringClause>
        </SimpleClause>
       </Clause>
      </LocalizedStringFilter>
     </NameBranch>
    </ClassificationNodeQuery>
  </ClassifiedByBranch>
       <SlotBranch> <SlotFilter> <Clause> <SimpleClause leftArgument = "name_">
              <StringClause stringPredicate = "Equal">originatingFrom</StringClause>
                          </SimpleClause> </Clause> </SlotFilter>
              <SlotValueFilter> <Clause> <SimpleClause leftArgument = "value">
              <StringClause stringPredicate = "Contains">Istanbul</StringClause>
                          </SimpleClause> </Clause> </SlotValueFilter>
       </SlotBranch>
       ...
 </ServiceQuery>  </FilterQuery> </AdhocQueryRequest>
```

Figure 7 An example ebXML query retrieving instances of a classification node

Finally, the retrieved services are depicted to the user and the user is allowed to form a workflow of selected services.

4.1 Implementation Status

A proof of concept implementation of the system is realized by using OASIS ebXML Registry Reference Implementation [ebXMLRR 2003]. As an application server to host Web services to be accessed through SOAP, Apache Tomcat 4.1 [Tomcat 2003] is used. The WSDL descriptions of the implemented services are generated through IBM Web services Toolkit 3.2 (WSTK) [WSTK 2000]. IBM Web Services Invocation Framework 1.0 (WSIF) [WSIF 2001] is used to invoke services.

5 Conclusions

Web Services will transform the web from a collection of information into a distributed device of computation. In order to employ their full potential, appropriate semantic mechanisms for web services need to be developed. In this paper we describe a semantic-based service composition tool for ebXML registries.

Service discovery and composition by using the semantics of services has been addressed in the literature where the weight of the work has been on using AI techniques to match the

inputs and outputs of services requested and advertised. The main problem with these approaches is the performance, which is not yet at the levels to be adopted by the industry.

In contrast to such approaches, we take a data management approach for service discovery by exploiting the meta data and the query mechanism of the ebXML registry. With the tools we have provided, the registry is queried for explicit members of a classification node to obtain the generic parameters of the service and users are allowed to fill these in through graphical user interfaces generated dynamically. Then the registry is queried for matching services; the corresponding WSDL files are obtained and through a workflow mechanism presented, the users can compose the discovered services. The current workflow mechanism implemented is a simple one realizing basic workflow functionality. As a future work, we plan to use one of the recent Web services choreography languages, such as BPEL4WS [BPEL4WS 2002] or WSCI [WSCI 2002] in composing the services.

References

BPEL4WS: Business Process Execution Language for Web Services, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Bussler C., Fensel D., Payne T., Sycara K.: "ISWC Tutorial: Semantic Web Services", http://www.daml.ri.cmu.edu/tutorial/iswc-t3.html\#2b

ebRIM: ebXML Registry Information Model v2.1, June 2002, http://www.ebxml.org/ specs/ebRIM.pdf.

ebRSS: ebXML Registry Services Specification v2.1, June 2002, http://www.ebxml.org /specs/ebiRS.pdf.

ebXML: http://www.ebxml.org/

ebXMLRR: OASIS ebXML Registry Reference Implementation Project (ebxmlrr), http://ebxmlrr.sourceforge.net/

SOAP: Simple Object Access Protocol, http://www.w3.org/TR/SOAP/

Tomcat: http://jakarta.apache.org/.

UDDI: Universal Description, Discovery and Integration, http://www.uddi.org

UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/

WSCI: Web Service Choreography Interface, http://wwws.sun.com/software/xml/developers/wsci/

WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl

WSIF: IBM Web Services Invocation Framework, http://www.alphaworks.ibm.com/tech/wsif

WSTK: IBM Web Services Toolkit (WSTK), http://alphaworks.ibm.com/tech/webservicestoolkit

PROJE ÖZET BİLGİ FORMU

Proje Kodu : 101E041

Proje Başlığı :
UDDI Depoları için Sanal İş akışı desteği (Virtual Enterprise Support for UDDI Registries)

Proje Yürütücüsü ve Yardımcı Araştırıcılar :
Prof. Dr. Asuman Doğaç
Yıldıray Kabak
Gökçe Banu Laleci

Projenin Yürütüldüğü Kuruluş ve Adresi :
Yazılım Araştırma ve Geliştirme Merkezi (Software Research & Development Center)
İnönü Bulvarı, SRDC, Bilgisayar Müh. Bölümü, ODTÜ

Destekleyen Kuruluş(ların) Adı ve Adresi :

Projenin Başlangıç ve Bitiş Tarihleri : 01/03/2002 - 01/03/2003

Öz : (en çok 70 kelime)
Projede elektronik ticaret konusunda günümüz teknolojisi olan UDDI kullanımının yaygınlaştırılması ve kolaylaştırılması için bir otomasyon sistemi geliştirmek amaçlanmaktaydı. Bu amaçla şirketlerin ürün ve fiyat bilgilerine ulaşabilmek için bir yöntem geliştirilmesi, ve bu yöntem sayesinde kullanıcıdan alınan bilgilerle hangi şirketin hangi servislerinin kullanılacağının otomatik olarak bulunup bir iş akışı sistemi sayesinde uygun sıraya konup insan müdahalesi gerektirmeden çalıştırılabilmesi amaçlanmaktaydı. Proje kapsamında bu sistem yazılımı sayesinde internet üzerinden herhangi bir şirket otomatik olarak kendine iş ortağı bulabilmektedir.

Anahtar Kelimeler:
UDDI, WSDL, SOAP, Virtual Enterprises

Projeden Kaynaklanan Yayınlar :

1. Dogac, A., Laleci, G., Kabak, Y., Cingil, I.,"Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002, http://www.research.microsoft.com/research/db/debull/issues-list.htm.
2. Dogac, A., Cingil, I., Laleci, G. B., Kabak, Y., "Improving the Functionality of UDDI Registries through Web Service Semantics", 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002.
3. Dogac, A., Tambag, Y., Pembecioglu, P., Pektas, S., Laleci, G. B., Kurt, G., Toprak, S., Kabak, Y., "An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs", ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 2002.
4. Dogac, A., Laleci, G., Kurt, G., Kabak, Y., Acar, A., "A Platform for Semantically Enriched Mobile Services" , in Proc. of the First International Conference on Mobile Business, Athens,

Greece, July 2002.
5. Dogac, A., Tumer, A., "Issues in Mobile Electronic Commerce", Journal of Database Management, Vol. 13, No. 1, January 2002, pp. 37-43.
6. Dogac, A., Guest Editor, ACM Sigmod Record, Special Section on Data Management Issues in e-commerce, Vol. 31, No. 1, March 2002.
7. Ozen, B., Kilic, O., Altinel, M., Dogac, A. "Highly Personalized Information Delivery to Mobile Clients", KLUWER, Wireless Networks, the Journal of Mobile Communication, Computation and Information.
8. Dogac, A., Kabak, Y., Laleci, G., "A Semantic-Based Web Service Composition Facility for ebXML Registries", ICE2003 9th International Conference on Concurrent Enterprising, Helsinki, Finland, June 2003.

Bilim Dalı :
Doçentlik B. Dalı Kodu :