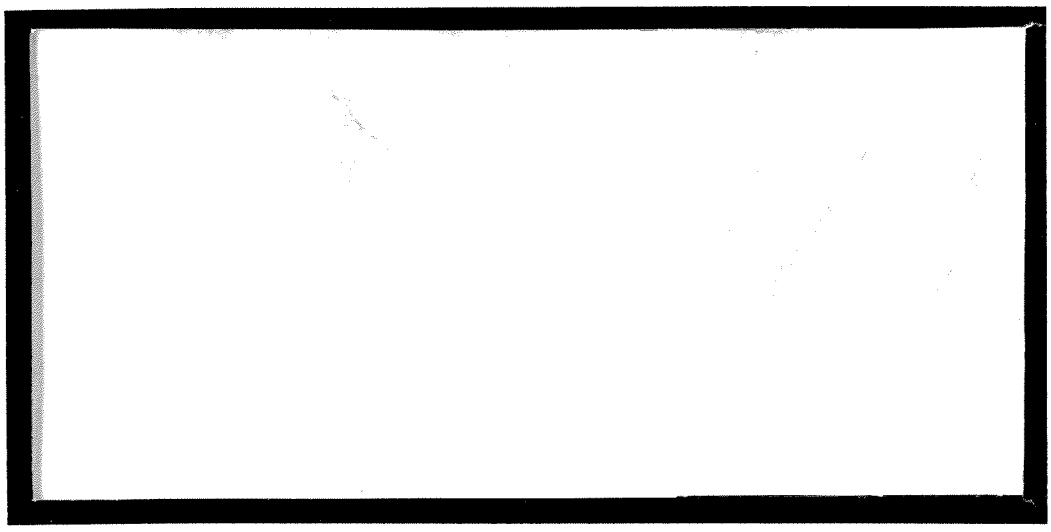
DVP

2004-263

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

ANLAMSAL OLARAK ZENGİNLEŞTİRİLMİŞ MOBİL SERVİSLER PLATFORMU

(A PLATFORM FOR SEMANTICALLY ENRICHED MOBILE SERVICES)

PROJE NO : 102E035

PROF.DR. ASUMAN DOĞAÇ

Mart 2004

ANKARA

ÖNSÖZ

Bu projede taşınabilir cihazlardan kullanıcıların İnternet aracılığıyla ticaret yapabilmesini sağlamak amacıyla bir alt yapı geliştirilmiştir. Bu amaçla, web servislerin bilgilerine, bu servisler için tanımlanmış semantik aracılığıyla ulaşabilmek için bir yöntem geliştirilmiş; böylece servis isteyicisinden alınacak bilgiyle, hangi servislerin kullanılacağına, insan müdahalesi minimum seviyede tutularak, ajanlar tarafından karar verilip düzenlenmesi gerçekleştirilmiştir.

Proje kullanıcıların "Güvenilir otoriteler" aracılığıyla kendilerini ortama tanıtmaları ve JADE ve LEAP platformlarında geliştirilmiş olan ve değişik işlevleri bulunan ajanların işbirliğiyle kullanıcının ihtiyaçlarına uygun web servislerin bulunması ve kendilerine ulaştırılmasını sağlamaktadır. Bu projede yapılan çalışmalar sonucu elde edilen bilgilerin ve yazılımların kurumumuz bünyesinde devam eden Avrupa projelerinde kullanılması amaçlanmaktadır. Geliştirilmiş olan yazılımın bu projelerdeki endüstriyel ortakların sistemlerine entegre edilerek uygulamaya aktarılması tasarlanmaktadır.

Aynı zamanda bu projede elde ettiğimiz bilgilerin de büyük yardımıyla Avrupa Komisyonu 6. Çerçeve Programına sunmuş olduğumuz ve Koordinatörü oldugumuz aşağıdaki iki projemiz Avrupa Komisyonu degerlendirmelerinde eşik değerini geçmiş olup, Satine isimli projemiz STREP grubunda 3. , Artemis isimli projemiz ise STREPler icinde 5. olmuştur. Her iki projemiz de desteklenmek için görüşmeye çağrılmış ve 1 Ocak 2004 tarihinde başlamış olup kurumumuz tarafından yürütülmektedir.

1- ARTEMIS: A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information systems
2- SATINE: Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer Networks

Bu proje TÜBİTAK EEEAG (Elektrik Elektronik ve Enformatik Araştırma Grubu) tarafından desteklenmiştir.

İÇİNDEKİLER

ÖNSÖZ....2
İÇİNDEKİLER....3
ÖZ....4
ABSTRACT....5
PROJE ANA METNİ....6
GİRİŞ....6
GENEL BİLGİLER....6
YÖNTEM....7
BULGULAR....8
TARTIŞMA/SONUÇ....11
PROJE ÖZET BİLGİ FORMU....12
REFERANSLAR....13

ÖZ

Mobil-Pazar uygulamaları pazarlaması yavaş ilerlemekte ve bu tür servislerin yayılması ekran koruyucusu, zil sesleri indirmek ya da son haberlere ulaşmak gibi basit bilgi-eğlence servisleri ile sınırlı kalmıştır. Bunun bazı sebebleri pazarlama ile ilgili olsa da gerçek etkileşimli Mobil-Pazar kullanıcı deneyimlerinin yayılması altında zorlu birtakım teknik sorunlar yatmaktadır. Bunların başında:

1. Uygun, tutarlı ve güvenilir bir şekilde kullanıcı kontekstinin yer, servis teklifleri, diğer kullanıcılar ve ortam değişkenleri yönünden elde etmek.
2. Güven ve gizlilik servisleri/bilgilerinin yönetimi kullanıcılara bilgilerin korunduğunu fakat kullanıcının yararı için gerektiğinde kullanılacağı garantisini vermek.
3. Kullanıcıların servis keşfi, erişimi, düzenlemesi, ve ayrıca servis semantiğinin birçok servis etkileşimi gerektiren karmaşık hedeflerin başarılmasının etkili yönetim ve kontrolü.

Her bir alandaki başarı izsiz, tutarlı, her yerde olan servislerin tedariği açısından kritiktir. Ayrıca çözümler kullanıcının elindeki cihazdaki yazılımdan ağa yayılan servisler kadar ağın birçok aşamasında geliştirilmelidir.
Mobil cihazlar için ajan teknolojileri, ajan iletişimi, bilgi gösterilmesi için diller, web servis semantiği, web servis depoları, güvenlik ve gizlilikle ilgili web standartlarındaki zorlukları aşabilecek geleceği parlak birçok teknoloji ve başlangıçlar varken, bunların nasıl uygun bir şekilde birleştirileceği, yayılacağı ve gerçek bir üretim ortamında kullanılacağı açık değildir.

Telesto prejesi yukarıdaki problemlere yönelik dikey entegre edilmiş, üretime hazır bir gösterim ortamı geliştirecek endüstri ve araştırma kuruluşlarını biraraya getirmeyi amaçlamaktadır.

Özellikle eskiden insan etkileşimi gerektiren angaryaları otomatikleştirmek için ajan teknolojisinin, web servislerin ajanlar için anlaşılabilir kılınması için semantik web teknolojisinin ve akıllı ajanlar vasıtasıyla web servis keşfini, düzenleme ve çalıştırılmasını otomatikleştirmek için de web servislerin kilit bir rol oynadıklarına inanıyoruz. Proje, var olan tanımlamaların adaptasyonu ve yenilerinin geliştirilmesi için temel ve uygulamalı araştırma, gösteri ortamının kurulması için geliştirme ve ve yayma, ortaya çıkan sistemi denemek için önemli alan denemeleri, sonuçların gerçek kullanıma transferini sağlamak için tedarikleri el almayı içerecektir.

Anahtar kelimeler: FIPA, JADE, LEAP, ebXML, OKBC, Web Service

ABSTRACT

Innovating new M-commerce applications have been slow in coming to market and deployments so far have been primarily limited to simple infotainment services such as downloading screen savers and ring tones, or getting the latest news. While some of the reasons for this are market related there are also a number of very challenging technical problems that underlie the deployment of true end-to-end M-commerce user experiences. Chief amongst these are:

4. Coherent, consistent and reliable capture of user context in terms of location, service offerings, other users, environmental conditions.
5. The management of trust and privacy services/information ensuring users data is protected yet can be leveraged to the user's benefit where appropriate.
6. Effective management and control of user service discovery, access, composition and hence service semantics to achieve complex goals requiring multiple service interactions.

Success in each of these areas is critical to the provision of seamless, consistent, ubiquitous service offerings. Furthermore solutions require developments at many levels of the network from the software in the users handset to services deployed in the network

While there are a considerable number of promising technologies and initiatives which may help address these challenges such as agent technologies for mobile devices, agent negotiation, Markup languages for knowledge representation, Web service semantics, Web service registries, Web standards related with security and privacy; it is unclear how they can be adequately combined, deployed and used in a real production environment. The Telesto project aims to bring together industry and research organizations to develop a vertically integrated, production-ready demonstration environment that addresses the problems outlined above.

In particular we believe that a key role will be played by: agent technology to automate the chores that previously required human attention, semantic web technology to make Web services understandable to agents and web services to automate Web service discovery, composition and execution through intelligent agents. The project will involve: core and applied research for adaptation of existing formalisms and the development of new ones, development and deployment to create the demonstration environment, a significant field trial to test the resulting system, take up provisions to enable the transfer of results into real usage.

Keywords: FIPA, JADE, LEAP, ebXML, OKBC, Web Service

PROJE ANA METNİ

GİRİŞ

Web servislerinin geleceğine bakıldığında, ajanların asıl ani değişikliği web servislerinin tümünü, kullanıcının ihtiyaç duyduğu zaman internette gezip, arayıp bulmak zorunda olduğu hareketsiz, yığın halindeki bilgiden, mobil kullanıcının ihtiyaçlarını onun profil ve genel durumuna bağlı olarak sağlayacak dinamik bir yetenekler kümesine çevirmesiyle yapacağı düşünülünüyor. Bu değişiklik şu zorlukları içermektedir: Öncelikle web servis keşfi ve yürütmesi son kullanıcının kullanım rahatlığı için otomatikleştirilmelidir. Bu ise servislerin anlamsal olarak betimlenmesini ve kullanıcının içersinde bulunduğu genel durumu - sadece bulunduğu yer açısından değil - yakalamayı gerektirmektedir. Bu kullanım rahatlığı için gerekli olan, akıllı ajanların web servislerini kullanıcının ihtiyaç ve tercihlerine göre araştırmaları ve birleştirmeleridir. Son olarak, web servislerinin mobil cihazlara erişiminin yazılımsal ajanlarla erişilebilmelerini sağlamak için, ajanların sınırları da göz önüne alınmalıdır. Biz anlamsal olarak betimlenmiş web servislerinin mobil cihazlar ile de erişilebilmesini ajan teknolojisi ve servis saklayıcıları aracılığıyla sağlamak amacıyla bir altyapı tanımlıyoruz. İlgili standartlarla olan ihtiyacı ve alakalarını tanımlayıp, araştırmaya açık problemleri araştırdık.

GENEL BİLGİLER

Son zamanlarda web semantiğini anlatan çabalar bir ivme kazandı. Semantik Web olarak adlandırılan web bir sonraki kuşağı bilgisayarların otomatikleştirilmiş muhakemeyi yönetebilmeleri için yapılandırılmış bilgi yığınlarına ve çıkarım kuralları kümelerine erişim sağlama amacı içersindedir. Bu yöndeki önemli bir çaba özelleşmiş sınıflandırılmaları ve kaynakların özelliklerini anlatmaya yarayan bir dil olan DAML-OIL'dir. DAML-S, DAML-OIL temelinde oluşturulmuştur.

FIPA önceden belirlenmiş standartlara uyan, bahsedilen temellerin altyapısı olarak kullanılabilecek bir ajan üretim platformudur. FIPA ajanlar arası işlevsellik ontolojisi OKBC (Open KnowledgeBase Connectivity) aracılığı ile ontoloji sunucuları ile haberleşen ontoloji ajanları aracılığıyla desteklemektedir. Bu ortak bir dilde konuşan ajanlar eğer ortak alan kavramlarını simgelemek için farklı sözcük dağarcıkları kullanırlarsa yine birbirlerini anlayamayacakları için gereklidir. OKBC, KIF'e dayanmaktadır ve DAML-OIL ontolojileri tarafından sağlanan özelleşmiş sınıflandırma ve kaynak özelliklerini destekleyecek şekilde genişletilmesi gerekmektedir.

JADE, Telecom Italia Lab tarafından geliştirilen, FIPA ya uyan bir multi-ajan geliştirme platformudur. JADE in ana amacı, ajan tabanlı sistemlerin geliştirilmesini kolaylaştırmak ve bazı sistem servisleri ve ajanları yoluyla, FIPA standartlarına uyumu sağlamaktır.

LEAP Mobil araçlar için, açık altyapılara ve dinamik,mobil girişimlere olan ihtiyacı karşılar. LEAP, mobil girişim işgücünün üç ihtiyacını destekleyen ajan tabanlı servisler geliştirir. Bu üç ihtiyaç : Bilgi yönetimi (bireysel bilgi gereksinimlerini tahmin etmek), merkezden dağıtılmış iş koordinasyonu (bireyleri güçlendirmek, işleri koordine ederek alıp-vermek) ve seyahat yönetimidir (bireysel seyahat ihtiyaçlarını planlama ve koordine etme). Bu ajan tabanlı servislerin merkezinde, standartlaşmış bir ajan platformuna ihtiyaç vardır. LEAP, PDA ler ve mobil telefonlar gibi küçük aletlerde çalıştırılabilen, boyutu ve fonksiyonu geliştirilebilen, işletim sistemi agnostik bir ajan platformu geliştirmektedir.

EbXML "saklayıcısı" "depo"yu idare etmek için ve ticaret yapan ortaklar arasındaki bilgi paylaşımını sağlamak için bir takım servisler sunar. Ayrıca sınıflandırma ağaçları kurmaya ve "saklayıcı"daki objeleri sınıflandırma objeleri aracılıyla sınıflandırma tasarılarıyla ilişkilendirmeye izin verir.

YÖNTEM

Proje süresince yürütülecek yönteme göre, ilk etapta geliştirilecek olan yazılım parçalarının tasarımı yapılacaktı. Bu tasarım çerçevesinde sistemin temel özelliklerini taşıyan basit bir prototip yazılımı gerçekleştirilecek, bu prototip irdelenerek tasarımdaki eksiklikler giderilecek ve tasarım geliştirilecekti. Daha sonra bu tasarım çerçevesinde sistemin bütün özelliklerini içeren bir prototip yazılımı gerçekleştirilip, proje süresi boyunca literatür düzenli olarak takip edilerek literatürdeki son gelişmeler çerçevesinde tasarımda gerekli düzenleme ve değişiklikler yapılıp bu değişiklikler geliştirilecek olan prototiplere aktarılacaktı.

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

Başlıca Aşamalar	Ayrıntılı Bilgi	Zamanlama
Web servis tanımlarının genişletilmesi	Şu an literatürde bulunan web servis tanımlarının eksiklikleri çıkarılacak ve bu eksiklileri gidermek için tanımlar genişletilecektir.	1-2
Sunulan servis ile tanımlanmış semantik arasındaki ilişkiyi kurmak	Servis kaydındaki mekanizmalar incelenerek, sunulan servis ile tanımlanmış semantik arasındaki ilişki kurulacaktır.	3-4

Standart sorguların yazılması	Ontoloji tanımlarından gerekli bilgileri çekmek için standartlaşmış sorgular yazılacaktır.	3-4
Kullanıcı ajanlarının geliştirilmesi	Bu servisleri kullanacak ajanların, JADE ve LEAP ajan geliştirme platformunda geliştirilecektir.	5-10
OKBC'nin genişletilmesi ve Ontoloji ajanının geliştirilmesi	OKBC'nin DAML+OIL'e uygun olması için genişletilecek ve Ontoloji ajanı geliştirilecektir.	9-10
Güvenilir otoritenin ve akıllı kullanıcı arayüzü ajanlarının geliştirilmesi	Kullanıcı profillerinin depolanacağı güvenilir otorite ve kullanıcı arayüzü ajanları geliştirilecektir.	9-10
Tüm sistemin entegrasyonu	Geliştirilen tüm komponentlerin entegrasyonu gerçekleştirilecektir.	11-12

BULGULAR

1) *Web servis tanımlarının genişletilmesi:* Bu iş tabloda belirtildiği gibi projenin ilk iki ayını kapsamaktadır. Bu iş başarıyla tamamlanmıştır. Literatürde bulunan tüm web servisler bulunup güncellenmiştir.

Detaylı bir araştırma sonucunda yeni web servisleri tespit edilip tümü literatüre eklenmiştir. Yapılan çalışmalarla web servislerin tanımları detaylı bir şekilde irdelenmiş ve eksiklikleri bulunmuştur. Eksikliklerin tespitinden sonra bu eksiklikleri gidermek için web servislerin tanımlari geliştirilerek ve genişletilerek en uygun ve en güncel hale getirilmiştir.

2) *Sunulan servis ile tanımlanmış semantik arasındaki ilişkiyi kurmak:* Bu iş tabloda belirtildiği gibi projenin 3. ve 4 aylarını kapsayıp başarıyla tamamlanmıştır.

Tüm web servislerinin kayıtlarını ve kayıtlardaki mekanizmalarını inceleyerek bu servisler için tanımlanmış semantikler arasında karşılaştırmalı bir şekilde bir ilişki kurulması için yoğun çalışmalar yapılmıştır. Her bir web servis ayrı ayrı ve disiplinli bir şekilde proje ekibi tarafından incelenmiştir. Yapılan incelemeler ve edinilen tecrübeler ekip içerisinde periyodik olarak paylaşılmış ve projenin hızlandırılması için ortak bir çalışma içerisine girilmiştir. Bu iş ile eş zamanlı olarak yürütülmekte olan ve aynı süre zarfını kapsayan "Standart sorguların yazılması" işi de proje ekibinin iki gruba ayrılıp grupların işbölümü ile aynı zamanda başarıyla yürütülmüştür.

İncelemeler sonucunda her bir web servisin semantiği ile arasındaki ilişki tespit edilmiştir.

3) *Standart sorguların yazılması:* Tabloda da görüldüğü gibi projenin 3. ve 4. aylarını kapsayan bu iş başarıyla tamamlanmıştır.

Proje ekibinin bu iş ile ilgilenen alt grubunun özverili ve kapsamlı çalışmalarıyla bu iş sürdürülmüştür. Standartlaşmış sorguların oluşturulması safhasına geçilmeden ortak bir standart üzerine karar verilmiştir. Ontoloji tanımları itina ile incelenmiş ve hangi bilgilerin gerekli olduğu üzerine ortak bir karar alınmıştır. Gerekli görülen bilgilerin çıkartılamsıyla birlikte bu bilgilere erişim yolları üzerinde düşünülmüş ve ortaya verimli ve kullanışlı bir yol haritası konulmuştur. Bu metod kullanılarak proje ekibinin bu işten sorumlu grubu tarafından standartlaşmış sorgular tek tek yazılmış ve her birinin yeterliliği ve işlevliği test aşamasından geçirilerek performansları irdelenmiştir. Eksik yönleri görülen standartlaşmış sorgular bu sorgulardan sorumlu grup elemanlarınca itinayla gözden geçirilmiş ve değişiklikler yapılarak ya da bu sorgu için daha etkili yöntemler tespit edilip kullanılarak yeniden yazılmıştır. Tüm standartlaşmış sorgular bu aşamalardan geçmiş olup, son halleri kusursuz olmakla beraber bu iş başarıyla tamamlanmıştır.

4) *Kullanıcı ajanlarının geliştirilmesi:* Tabloda da görüldüğü gibi "Kullanıcı ajanlarının geliştirilmesi" işi projenin 5. ayından itibaren başlayıp 10. aya kadar tamamlanmıştır.

Öncelikle LEAP ve JADE platformları gözden geçirilmiş ve ajan geliştirmek için kullanılan bu platformların bu işi en güvenli, verimli ve istenilen özelliklere uygun olarak nasıl yapılacağı üzerinde durulmuştur. LEAP platformu JADE tabanlı olup, işlem gücü ve sistem kaynakları kısıtlı olan cihazlarda ajanlar yaratabilmek için tanımlanmıştır.Ajanlar mobil kullanıcının profilini ve de genel durumu hakkında bilgi sahibi olacak şekilde geliştirilmiştir. Bu sayede web servislerini kullanıcının ihtiyaçlarına göre zekice düzenleyebilecekler hale getirilmişlerdir. Bu ajanlar birbirlerinden ve ortamdan bağımsız olarak çalışmayacakları için belli standardlara uymaları gerekmektedir.

Bu sebeple dört çeşit ajanın geliştirilmesi gerekmiştir. Bu ajanların herbiri vasıflarına ve işlevselliklerine göre ayrı bir isimle adlandırılmaktadır. Bunlar:

- Akıllı arayüzü ajanları
- Kullanıcı ajanları
- Wrapper ajanları ve
- Ontoloji ajanlarıdır.

Akıllı arayüzü ajanları, kullanıcı ile kullanıcı ajanı arasında kullanıcının sistem ile uyumunu kolaylaştırmak amacıyla bilgi alışverişi sağlamak üzere LEAP platformunda geliştirilmektedir. Akıllı arayüzü ajanları kullanıcının gereksinimlerini tespit eder. Kullanıcının gereksinimlerini tespit etmek için ise kullanıcıyla uyumlu ve sistemin kompleksliğini kullanıcıya yansıtmadan kullanıcıdan gerekli verileri almak için bir kullanıcı arabirimi oluşturulmuştur. Bu arabirim ile kullanıcıya kullanım kolaylıkları getirilmektedir. Güvenilir otoriteler ile de temas halinde bulunabilen bu ajanlar kullanıcının profiline güvenilir otoriteler aracılığıyla erişilmesi için de kullanılır.

Kullanıcı ajanları, JADE platformunda geliştirilebilmektedir. Bu ajanlar akıllı arabirim ajanları aracılığıyla kullanıcının kendisiyle, standartlara uyabilmek için ontoloji ajanlarıyla, kullanıcıya talep ettiği web servisini sağlamak ve bunun için gerekli kontextlere erişim amacıyla "Kontext Servis Sunucusuyla", ve kullanıcının ihtiyaçları, kontext ve profiline göre web servislerini kullanıcıya ulaştırabilmek için de "ebXML Deposu" ile iletişim içerisinde olmakta ve bunlar arasındaki bütünlüğü sağlamaktadır.

Wrapper ajanları, JADE platformunda geliştirilebilen ajanlardır. Kullanıcı ajanları, Ontoloji ajanları, ve ebXML Deposu ile temas halinde olan wrapper ajanlarının asıl işlevi ebXML deposu ile haberleşerek kullanıcıya web servisin ulaştırılmasını sağlamaktır.

Ontoloji ajanları da JADE platformunda geliştirilmektedir. Diğer ajanlarla iletişimi yanısıra ontoloji ajanları "Ontoloji Servis Sunucusu" ile temas halindedir. Bu ajanlar "Ontoloji Servis Sunucusu"ndan aldığı bilgiler doğrultusunda diğer ajanların FIPA standartlarına uygunluğunu sağlar ve denetlerler.

Bu işin "kullanıcı ajanları"nın geliştirilmesi için gerekli altyapı çalışmaları tamamlanmış ve ajanların çalışması için gereken tüm işlemler başarıyla tamamlanmıştır. Ayrıca "kullanıcı ajanları"nın çalışmalarını test etmek ve bu ajanların performanslarını irdelemek, ve sistemin genel çalışmasını da denetlemek amacıyla projenin tüm işlevliğini kapsayan bir ortam kurulmuş ve diğer ajanlarında "kullanıcı ajanları" ile etkileşim içerisinde olduğu göz önüne alınarak, başlangıç aşamasında diğer ajanlardan da kendilerine düşen görevi yerine getirebilmeleri amacıyla temel fonksiyonlarını yerine getiren prototipler oluşturulmuştur. Ancak diğer ajanlar projede kendilerine ayrılmış zamanlarda daha detaylı olarak ele alınmıştır. Bu test ortamı sayesinde bir yandan kullanıcı ajanlarını geliştirirken eş zamanlı olarak çalışmaları da denetlenebilmektedir. Bu iş tüm ciddiyeti ve proje ekibinin özverisiyle zamanında tamamlanmıştır.

5) *OKBC'nin genişletilmesi ve Ontoloji ajanının geliştirilmesi:* OKBC DAML+OIL standartlarına uygun hale getirilmek için genişletilecek ve JADE platformunda Ontoloji ajanlarının geliştirilmesi tamamlanmıştır.

6) *Güvenilir otoritenin ve akıllı kullanıcı arayüzü ajanlarının geliştirilmesi:* Kullanıcının serbestliğini sağlayabilmek için güvenilir otorite geliştirilecek ve kullanıcıya kullanım kolaylığı sağlamak için "akıllı kullanıcı arayüzü ajanları"nın tasarımı yapılıp geliştirilmiştir.

Projede ayrıca Mesut Bahadır'ın tez çalışmasından faydalanılmıştır. Tez kapsamında taşınabilir cihazlardan Web servislerin çağırılması ele alınmıstır. Bu tezin projeyle ilgili kısmı ve altyapısı hakkında aşağıda kısaca bilgi bulunmaktadır.

SOAP her Web Servis uygulamasının temel taşıdır. Taşınabilir cihazlara SOAP/XML uygulaması geliştirmenin bir takım sorunları bulunmaktadır. Bu sorunların başında, günümüzde yaygın olarak kullanılmakta olan XML ve SOAP kütüphanelerinin oldukça büyük olması ve yüzlerce class bulundurmasıdır. Buna bağlı diğer bir sorun da, bu kütüphanelerin taşınabilir cihazlarda bulunmayan Java ortamı özelliklerine bağımlı olmasıdır. Neredeyse normalde J2EE ve J2SE sürümlerinde bulunan tüm Java çekirdek classlarından tamamen farklı Bağlı Sınırlı Cıhaz Konfigürasyonunu (Connected Limited Device Configuration) (CLDC) özellikle göz önüne alalım: AWT, Swing, Beans, Reflection, ve birçok java.util ve java.io classı basitçe ortadan kalkmıştır. Bu yalın Java ortamı iskeletinin amacı, taşınabilir cihazlarda çalışan düşük hafıza gereksinimli sanal makina KVM'nin sınırlılık özelliği ile bağdaştırmaktadır.

kSOAP kXML esaslı olup SOAP API'sinin Java 2'nin Taşınabilir Cihazlar için sürümü olan J2ME'ye uygunudur. Ufak çaplı olmasından dolayı, SOAP destekli Java Appletlerinin tanınmasına da uygun olabilir. kXML projesi J2ME (CLDC/MIDP/CDC) dahil tüm Java sürümlerine uygun kısıtlı bir XML işleyicisi ve yazıcısı sunmaktadır. Küçük boyutundan dolayı, özellikle Appletler ve Palm sistemleri ya da MIDP destekli cep telefonları gibi taşınabilir cihazlarda çalışan Java uygulamaları için uygundur. kXML ilk olarak Dortmund Üniversitesinin yapay zeka (AI) ünitesinde COMRIS projesinin bir 'yan ürünü' olarak geliştirilmiştir. API COMRIS projesinde, Bilgi Tabakası Modül veya Modüllerinde XMLleşmiş FIPA mesajlarını işlenmesinde ve şablon temelli XHTML sayfalarının oluşturulmasında kullanılmıştır. Şablon temelli XHTML sayfası oluşturma işlemi günümüzde Mlnet eğitim servis sağlayıcıları tarafından da kullanılmaktadır. kXML ayrıca Enhydra kXML-RPC ve kSOAP... projelerinde kullanılmaktadır.

7) *Tüm sistemin entegrasyonu:* Bu aşamada daha önce yapılmış diğer işlerin entegrasyonu yapılmıştır.

TARTIŞMA/SONUÇ

Proje başarıyla tamamlanmıştır. Geliştirilen sistem ile kullanıcılar istedikleri bir cihazdan (PDA, mobil telefon, vs.) akıllı kullanıcı arayüzü aracılığıyla kendi kullanıcı profillerini bir defa oluşturduktan sonra, kendilerine uygun bir güvenilir otorite bulup ona bağlanmakta. Güvenilir otorite ile kullanıcı kendi kullanıcı profilini paylaşır. Servis isteyicisinden alınacak bilgiyle, hangi servislerin kullanılacağına, kullanıcının müdahalesi minimum seviyede tutularak, ajanlar tarafından karar verilip düzenlenmesi gerçekleşmektedir. Böylece mobil cihazlarla internet üzerinden ticaret gerçekleştirilebilir hale getirilmiş ve kolaylaşmıştır. Bu projede yapılan çalışmalar sonucu elde edilen bilgiler ve yazılımlar kurumumuz bünyesinde devam eden Avrupa projelerinde kullanılmıştır.

Proje öneri formunda belirtilen amaç ve kapsama uygun olarak sonuçlanmış, Projede hiçbir farklılık olmamıştır.

PROJE ÖZET BİLGİ FORMU

Proje Kodu : 102E035

Proje Başlığı :
ANLAMSAL OLARAK ZENGİNLEŞTİRİLMİŞ MOBİL SERVİSLER PLATFORMU
(A PLATFORM FOR SEMANTICALLY ENRICHED MOBILE SERVICES)

Proje Yürütücüsü ve Yardımcı Araştırıcılar :
Prof. Dr. Asuman DOĞAÇ

Projenin Yürütüldüğü Kuruluş ve Adresi :
Yazılım Araştırma ve Geliştirme Merkezi (Software Research and Development Center)
İnönü Bulvarı, SRDC, Bilgisayar Mühendisliği Bölümü, ODTÜ
06531 Ankara TÜRKİYE

Destekleyen Kuruluş(ların) Adı ve Adresi :

Projenin Başlangıç ve Bitiş Tarihleri : 01/03/2003 – 01/03/2004

Öz : (en çok 70 kelime)

Proje farklı işlevleri bulunan ajanlar vasıtasıyla internet üzerinden ticareti kolaylaştırmayı amaçlamaktadır. Bu ajanlar kullanıcının ilgileneceği web servisleri bulup bunları kullanıcıya sunmakta. Bunu sağlamak için ajanların birbiriyle haberleşmesini sağlamak ve web servislerin ajanların anlayabileceği bir semantik ile tanıtılmasını ve servis ile bu semantik arasında bir bağlantı kurulması gerekmektedir. Proje kapsamındaki sisteminin yazılımı sayesinde kullanıcı profilini göz önünde bulunduran işlevsel ajanlar aracılığıyla kullanıcının internet üzerinden ticaret yaparken müdahalesi minimum seviyede tutulmaktadır.

Anahtar Kelimeler:
FIPA, JADE, LEAP, ebXML, OKBC, Web Service

Projeden Kaynaklanan Yayınlar :

Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", 14th International Workshop on Research Issues on Data Engineering, Boston, USA , March 28-29, 2004.

Dogac, A., Laleci, G., Kabak, Y.,"Context Frameworks for Ambient Intelligence", eChallenges 2003, Bologna, Italy, October 2003.

Dogac, A., "A Tutorial on Exploiting Semantic of Web Services through ebXML Registries", eChallenges 2003, Bologna, Italy, October 2003.
Bilim Dalı : **Doçentlik B. Dalı Kodu :**

REFERANSLAR

[DAML] DARPA Agent Markup Language, http://www.xml.com/pub/a/2002/01/30/daml1.html

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

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

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

[JADE] JADE, *http://sharon.cselt.it/projects/jade/*

[KIF] Finin, T., Labrou, Y., Mayfield, "A Brief Introduction to Knowledge Interchange Format", http://www.cs.umbc.edu/kse/kif/kif101.shtml

[LEAP] Lightweight Extensible Agent Platform (LEAP), European Commission's 5th Framework Project, ST-1999-10211, *http://leap.crm-paris.com/*

[W3C] World Wide Web Consortium, *http://www.w3.org/*

[WebOnt] World Wide Web Consortium, Requirements for a Web Ontology Language, http://www.w3.org/TR/2002/WD-webont-req-20020307

Asuman

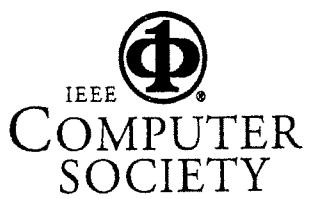
14th International Workshop on

Research Issues on Data Engineering:

Web Services for E-Commerce and E-Government Applications

Edited by
Athman Bouguettaya
Boualem Benatallah

Boston, Massachusetts
28-29 March 2004



Sponsored by
IEEE Computer Society Technical Committee on Data Engineering (TCDE)

Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery *

Asuman Dogac Yildiray Kabak Gokce B. Laleci
Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Türkiye
email: asuman@srdc.metu.edu.tr

Abstract

Web services, like their real life counterparts have several properties and thus truly useful semantic information can only be defined through standard ontology languages. Semantic Web is an important initiative in this respect. However, although service registries are the major mechanisms to discover services, the semantic support provided by service registries is completely detached from the Semantic Web effort.

In this paper, we address how ebXML registries can be enriched through OWL ontologies to describe Web service semantics. We describe how the various constructs of OWL can be mapped to ebXML classification hierarchies and show how the stored semantics can be queried through standardized queries by using the ebXML query facility.

We also provide our observations on how ebXML registries can be extended to provide more efficient semantic support.

1. Introduction

The efforts by industry in providing semantic support in service registries are disconnected from the Semantic Web initiative. UDDI registries use tModels to represent compliance with a taxonomy such as Universal Standard Products and Services Classification [13] and ebXML registries use classification hierarchies to associate semantics with registry items. On the other hand, through separate efforts, Web service semantics is being defined through powerful ontology languages like DAML-S [1] and more recently through OWL-S [11].

* This work is supported by the European Commission, Project No: IST-1-002104-STP SATINE and by the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 102E035

For semantic information to be truly useful, it should be able to define the properties that a service might have such as the methods of charging and payment, the channels by which the service is requested and provided, constraints on temporal and spatial availability, service quality, security, trust and rights attached to a service.

Such semantic information can only be defined effectively through ontology languages and Web Ontology Language (OWL) [10] is very important initiative in this respect. The future release of DAML-S [1], which defines a core set of markup language constructs for describing the properties and capabilities of Web services, will be based upon the Ontology Web Language (OWL) and will be called OWL-S [11].

The need for extending the UDDI registries with semantic capabilities has been recognized and addressed in the literature [2, 3, 12]. In [12], DAML-S specific attributes such as *inputs, outputs* and *geographicRadius* are represented using tModel mechanisms of UDDI. [2] and [3] also address importing semantic to UDDI registries where a mechanism is proposed to relate DAML-S ontologies with services advertised in the UDDI registries.

However, to the best of our knowledge how to extend ebXML registries with ontology language constructs has not been addressed in the literature. Since the semantic support provided by UDDI and ebXML registries differ considerably, it is not possible to repeat the previous work in UDDI for ebXML.

In this paper, we show how ebXML registries can be enriched through OWL-S ontologies for describing the Web service semantics. We describe how the various constructs of OWL can be mapped to ebXML classification hierarchies and show how the services are discovered through standardized queries by using the ebXML query facility.

In this way, we believe a mutually beneficial relationship is created between ebXML registries and the OWL ontology language: the former gains the ability to store standard OWL ontologies and the mechanisms provided by the

0-7695-2095-2/04 $20.00 © 2004 IEEE

ebXML registry proves very useful in associating the semantics defined in an OWL ontology with services advertised in the registry.

In the following, we describe the issues to be addressed to make Web service semantics defined through ontology languages like OWL to be accessible from service registries by also indicating the differences between UDDI and ebXML registries:

- *Where to store the service ontology so that it can be accessible from service registries.* In UDDI registries there is no mechanism to store the ontology. Therefore generic ontology descriptions can be stored by the standard bodies which define them. And the server, where the service is defined, can host the semantic description of the service instance.

  ebXML registry, on the other hand, 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.

- *How to relate the generic ontology classes with the service instances advertised in registries.* In UDDI registries, a tModel describes a technical model for representing a reusable concept such as a taxonomy. By putting the tModel key in the category bag of the Web service, a certain semantic is introduced. For example when we see the tModel key corresponding to the UNSPSC code [13] "43.16.17.02.00" in the category bag of a Web service we realize that this service is about *Tax preparation software.*

  In enriching UDDI registries with DAML-S semantics, [12] represents DAML-S specific attributes such as *inputs, outputs* and *geographicRadius* by using tModel mechanisms of UDDI.

  ebXML semantic support mechanism is different: In an ebXML compliant registry, *Classification* objects relate registry objects (which includes Web services) with the nodes of a *ClassificationScheme.* That is, a *Classification* instance classifies a RegistryObject instance by referencing a node defined within a particular classification scheme [4]. It is possible to define a scheme mapping OWL constructs into ebXML registry mechanisms. The transformations are based on the semantic constructs of ebXML registry such as the class hierarchy and the predefined associations (Table 1). Since ebXML allows this list to be expanded, it is possible to add new associations to map OWL into ebXML as detailed in Section 2.3. Once the transformations are defined, it is possible to automate the process. In fact, we have developed a tool to create an ebXML Classification Hierarchy from a given OWL ontology automatically by using the transformations proposed in Section 2.3.

- *How to discover the services according to their semantics.* The mechanism in UDDI registries are the APIs which allow to search for service instances according to tModel keys. The difficulties with tModels are as follows: to use them, one has to know the existing tModels. Furthermore, it is not possible to express any relationship between tModels. Also when introducing a new ontology to an UDDI registry other than already existing ones, there is a need for a dictionary to keep track of the terms of the ontology and their corresponding tModels.

  On the other hand, it is possible to store and browse an ontology in an ebXML registry and to define different kinds of relationship between registry items. Furthermore there is a query facility to access the registry items according to their semantics which allows both to query the ontology and to find services instances according to the classes in the ontology. As an example, the explicit members of a given class that satisfies a certain constraint can be located through querying. How this is achieved is described in Section 3.5.

The paper is organized as follows: Section 2 describes how service semantics defined in OWL ontologies can be stored and accessed in ebXML registries. In Section 3 we describe how to use the defined semantics in locating services in the ebXML registries. Finally in Section 4, we state our observations on how ebXML registries can be extended to provide more efficient semantic support by also storing the semantics of individual service instances.

2. Storing OWL Ontologies into ebXML Registries

In order to describe how OWL ontologies can be stored in ebXML registries we first briefly summarize the semantic constructs they each provide.

2.1. Web Ontology Language (OWL)

OWL describes the *structure* of a domain in terms of *classes* and *properties*. Classes can be names (URIs) or *expressions* and the following set of *constructors* are provided for building class expressions: *owl:intersectionOf, owl:unionOf, owl:complementOf, owl:oneOf, owl:allValuesFrom, owl:someValuesFrom, owl:hasValue.*

In OWL, properties can have multiple domains and multiple ranges. Multiple domain (range) expressions restrict the domain (range) of a property to the intersection of the class expressions.

Another aspect of the language is the axioms supported. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties

Name	*Description*
RelatedTo	Defines that source RegistryObject is related to target RegistryObject.
HasMember	Defines that the source RegistryPackage object has the target RegistryObject object as a member.
ExternallyLinks	Defines that the source ExternalLink object externally links the target RegistryObject object.
Contains	Defines that source RegistryObject contains the target RegistryObject.
EquivalentTo	Defines that source RegistryObject is equivalent to the target RegistryObject.
Extends	Defines that source RegistryObject inherits from or specializes the target RegistryObject.
Implements	Defines that source RegistryObject implements the functionality defined by the target RegistryObject.
InstanceOf	Defines that source RegistryObject is an Instance of target RegistryObject.
Supersedes	Defines that the source RegistryObject supersedes the target RegistryObject.
Uses	Defines that the source RegistryObject uses the target RegistryObject in some manner.
Replaces	Defines that the source RegistryObject replaces the target RegistryObject in some manner.
SubmitterOf	Defines that the source Organization is the submitter of the target RegistryObject.
ResponsibleFor	Defines that the source Organization is responsible for the ongoing maintainence of the target RegistryObject.
OffersService	Defines that the source Organization object offers the target Service object as a service.

Table 1. Predefined Association Types in ebXML Registries

OWL	*ebXML*
owl:Class	ClassificationNode
rdf:Property	Association
rdfs:domain	sourceObject
rdfs:range	targetObject
owl:equivalentTo	An association with a predefined association type of "EquivalentTo".
rdfs:subClassOf	An association with a new association type "subClassOf" is defined.
owl:ObjectProperty	An association with a new association type "ObjectProperty" is defined.
owl:DataTypeProperty	XML Schema datatypes are used by providing an external link from the registry.
owl:sameClassAs	An association with a predefined association type of "EquivalentTo".
rdfs:subPropertyOf	An association between associations with a new association type "subPropertyOf" is defined.
owl:samePropertyAs	An association between associations with a predefined association type of "EquivalentTo" is defined.
owl:disjointWith	An association with a new association type of "disjointWith" is defined.
owl:sameIndividualAs	An association instance with a predefined association type of "EquivalentTo" is defined.
owl:differentIndividualFrom	An association instance with a predefined association type of "disjointWith" is defined.
owl:inverseOf	An association between associations with a new association type "inverseOf" is defined.
owl:TransitiveProperty	A slot is associated with the property to indicate that this property is transitive.
owl:FunctionalProperty	A slot is associated with the property to indicate that the subject which has this property uniquely identifies the object (value) of the property.
owl:InverseFunctionalProperty	A slot is associated with the property to indicate that the object (value) uniquely identifies the subject which has this property.
owl:SymmetricProperty	A slot is associated with the property to indicate that it is symmetric, that is, if the pair (x,y) is an instance of this property then (y,x) is also an instance of this property
owl:intersectionOf	A registry package is created by associating the classes (i.e. the classification nodes) to be intersected through a new association type of "intersectionOf".
owl:unionOf	A registry package is created by associating the classes (i.e. the classification nodes) whose union is to be taken, through a new association type of "unionOf".
owl:complementOf	An association with a new association type "complementOf" is defined.
rdf:parseType="Collection"	A RegistryPackage is created.
owl:oneOf	A registry package is created by associating the classes (i.e. the classification nodes) in the owl:collection by a new type of association "oneOf".
owl:Restriction	Since ebXML RIM associations have local scope, only the type of the Restriction needs to be specified.
owl:allValuesFrom	A slot type "allValuesFrom" is defined for the association representing a restriction.
owl:someValuesFrom	A slot type "someValuesFrom" is defined for the association representing a restriction.
owl:hasValue	A slot type "hasValue" is defined for the association representing a restriction.
owl:cardinality	An association with a new association type "cardinality" is defined.
owl:minCardinality	An association with a new association type "minCardinality" is defined.
owl:maxCardinality	An association with a new association type "maxCardinality" is defined.

Table 2. Mapping OWL Ontologies to ebXML Classification Hierarchies

[6]. The following are the set of OWL axioms: *rdfs:subClassOf, owl:sameClassAs, rdfs:subPropertyOf, owl:samePropertyAs, owl:disjointWith, owl:sameIndividualAs, owl:differentIndividualFrom, owl:inverseOf, owl:transitiveProperty, owl:functionalProperty, owl:inverseFunctionalProperty.*

2.2. ebXML Registry Information Model (RIM)

Figure 1 presents the part of the ebXML RIM [4] related with storing metadata information. The top level class in RIM is the "RegistryObject". This is an abstract base class used by most classes in the model. It provides minimal metadata for registry objects. "Slot" instances provide

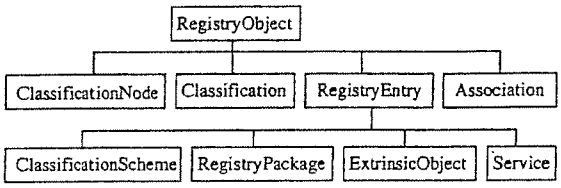


Figure 1. A Part of the ebXML RIM Class Hierarchy

a dynamic way to add arbitrary attributes to "RegistryObject" instances. "Association" instances are used to define many-to-many associations between objects in the information model. An Association instance represents an association between a source RegistryObject and a target RegistryObject. Each Association must have an "associationType" attribute that identifies the type of that association. There are a number of predefined *Association Types* that a registry must support to be ebXML compliant [4] as shown in Table 1. ebXML allows this list to be expanded.

"ClassificationScheme" instances describe a structured way to classify or categorize RegistryObject instances. A ClassificationScheme defines a tree structure made up of "ClassificationNode"s. RegistryPackage instances, on the other hand, group logically related RegistryObject instances together.

Classifications could be internal or external depending on whether the referenced classification scheme is stored internal to the registry or it is external. An internal classification always references the node directly by its id while an external classification references the node indirectly by specifying a representation of its value that is unique within the external classification scheme.

2.3. Representing OWL Ontologies through ebXML Classification Hierarchies

From the descriptions presented in the sections 2.1 and 2.2, it is clear that there are considerable differences between an OWL ontology and an ebXML class hierarchy in terms of semantic constructs. In this section, we explain how these differences can be accommodated.

OWL classes can be represented through "ClassificationNodes" and RDF properties that are used in OWL can be treated as "Associations". An "Association" instance represents an association between a "source RegistryObject" and a "target RegistryObject". Hence "rdfs:domain" property can be mapped to a "source RegistryObject" and "rdfs:range" can be mapped to a "target RegistryObject". Consider the following example which defines a property

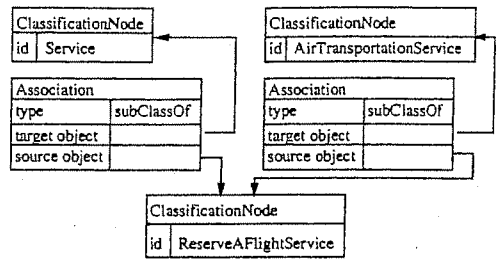


Figure 2. Representing an Example "owl:subClassOf" Property in ebXML Registry

"promotion" whose domain is the "TransportationService" and whose range is "PromotionRequirements":

```
<rdf:Property rdf:ID="promotion">
   <rdfs:domain rdf:resource="#TransportationService"/>
   <rdfs:range rdf:resource="#PromotionRequirements"/>
</rdf:Property>
```

In order to define this property in ebXML RIM, first, two classification nodes are created, namely "TransportationService" and "PromotionRequirements". Then an association, called "promotion" of type "Property" is defined where the "sourceObject" is "TransportationService" and the "targetObject" is "PromotionRequirements", as shown in the following:

```
<rim:ClassificationNode id = 'TransportationService'
parent= 'TravelService'> </rim:ClassificationNode>
<rim:ClassificationNode id = 'PromotionRequirements'
parent= 'TravelService'> </rim:ClassificationNode>
<rim:Association id = 'promotion'
associationType = 'Property' sourceObject =
'TransportationService' targetObject =
'PromotionRaquirements' >
</rim:Association>
```

A problem to be handled over here is that in OWL properties can have multiple domains and multiple ranges. Multiple domain (or range) expressions restrict the domain (or range) of a property to the intersection of the class expressions. The way to handle this in ebXML is to create a "RegistryPackage" for multiple domain (or range) classes and then take the intersection of these classes.

When it comes to mapping OWL class hierarchies to ebXML class hierarchies, OWL relies on RDF Schema for building class hierarchies through the use of "rdfs:subClassOf" property and allows multiple inheritance. An ebXML Class hierarchy has a tree structure, and therefore is not readily available to express multiple inheritance, that is, there is a need for additional mechanisms to express multiple inheritance. We define a "subClassOf" property as an association for this purpose.

Consider the example:

```
<owl:Class rdf:ID="ReserveAFlight">
<rdfs:subClassOf rdf:resource="http://www.daml.org/services/
                              daml-s/2001/05/Profile.owl"/>
<rdfs:subClassOf rdf:resource="#AirTransportationService"/>
</owl:Class>
```

Here, "ReserveAFlight" service inherits both from "AirTransportation" service and OWL-S ServiceProfile class. Figure 2 shows how this is represented through ebXML RIM constructs. Figure 3, on the other hand, demonstrates the snapshot of ebXML Registry Browser (the registry implementation given in [9] is used) showing this example.

Since OWL properties are represented through ebXML associations, "rdfs:subPropertyOf" is defined as an association between associations with a new association type of "subPropertyOf".

To express "owl:samePropertyAs", that is, the fact that the two properties are the same, an association of predefined ebXML type "EquivalentTo" is used between associations.

"owl:FunctionalProperty" asserts that a property can have only one value "y" for each instance of "x". As an example, the following definition indicates that "hasPrice" property can have only one corresponding value for each instance of "ReserveAFlight" service class.

```
<owl:FunctionalProperty rdf:ID="hasPrice">
  <rdfs:subpropertyOf rdf:resource="http://www.daml.org/
        services/daml-s/2001/05/Profile.owl"/>
  <rdfs:domain rdf:resource="#ReserveAFlight"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/
              XMLSchema/nonNegativeInteger"/>
</owl:FunctionalProperty>
```

To describe this semantics, we associate a slot with the association "hasPrice" to indicate that it has a unique value as shown in the following:

```
<rim:Association id = 'hasPrice'
associationType = 'objectProperty'
sourceObject = 'ReserveAFlight'
targetObject = 'nonNegativeIntegerDataType' >
              <Slot name = 'unique'>
              <ValueList>
                    <Value>true</Value>
              </ValueList>
</rim:Association>
```

OWL allows Boolean combinations of class expressions. For instance, "owl:complementOf" defines a class that consists of exactly all objects that do not belong to the class expression. Consider the following example:

```
<owl:Class rdf:ID="AirTransportation">
  <owl:complementOf rdf:resource="#LandTransportion"/>
</owl:Class>
```

To express "owl:complementOf" property, a new association of type "complementOf" is used between class nodes.

OWL allows to define enumerated sets through *rdf:parseType="Collection"*. ebXML "RegistryPackage" is used to express such collections to indicate that its elements should be treated as a unit. The class defined by "owl:oneOf" element, on the other hand, contains exactly the enumerated elements, no more, no less.

Consider the following example:

```
<owl:Class ID="Cities">
   <owl:oneOf rdf:parseType="Collection">
     <owl:Thing rdf:ID="Berlin"/>
     <owl:Thing rdf:ID="Istanbul"/>
   </owl:oneOf>
</owl:Class>
```

To express "owl:oneOf", we create a new association of type "oneOf" and associate the package itself, with every element in the package through this newly defined property.

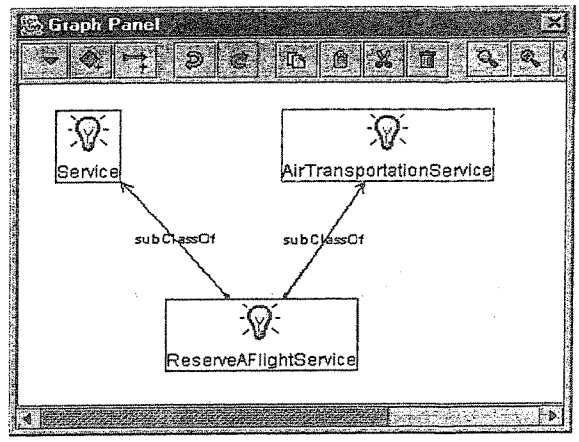


Figure 3. A Snapshot of ebXML Registry Browser Depicting the Example Class Hierarchy of Figure 4

Another important construct of OWL is "owl:Restriction". In RDF, a property has a global scope, that is, no matter what class the property is applied to, the range of the property is the same. "owl:Resriction", on the other hand, has a local scope; restriction is applied on the property within the scope of the class where it is defined. (The aim is to make ontologies more extendable and hence more reusable.) OWL provides the following language elements to indicate the type of restriction: *owl:allValuesFrom, owl:someValuesFrom, owl:-hasValue*. An *owl:allValuesFrom* element defines the class of all objects for whom the values of property all belong to the class expression.

Consider the following example:

```
<owl:Class rdf:ID="ReserveAFlight">
  <rdfs:subClassOf rdf:resource="&service"/>
  <rdfs:subClassOf rdf:resource=
   "#AirTransportationService"/>
  <rdfs:subClassOf>
    <owlRestriction>
```

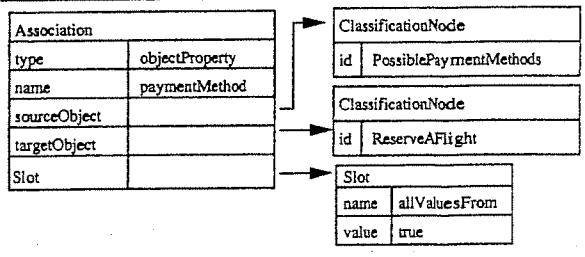


Figure 4. Representing an Example OWL Restriction in ebXML Registry

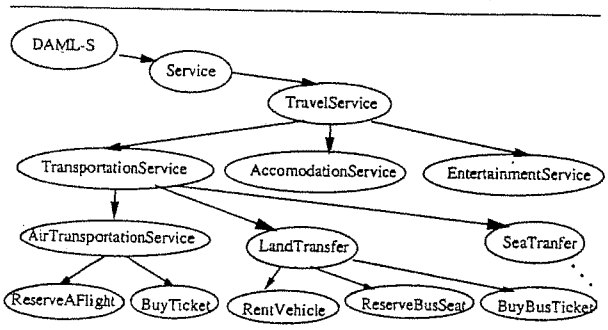


Figure 5. An Example Class Hierarchy for Travel Domain

```
    <owl:onProperty rdf:resource="#paymentMethod"/>
    <owl:allValuesFrom rdf:resource=
    "#PossiblePaymentMethods"/>
    </owl:Restriction> </rdfs:subClassOf>
</owl:Class>
```

Here "owl:Restriction" defines an anonymous class, that is the class of all things that satisfy this restriction. The restriction is that the property "paymentMethod" should get all of its values from the class "PossiblePaymentMethods". By defining "ReserveAFlight" service class as a subclass of this anonymous class, its "paymentMethod" property is restricted to the elements of the "PossiblePaymentMethods".

In ebXML class hierarchies, on the other hand, an association (which represents a property) is defined already in a local scope by associating two nodes of the class hierarchy. The type of the restriction can be expressed by special slot values. Figure 4 shows how the example above is represented through ebXML RIM constructs.

OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply including their URIs within an OWL ontology. In ebXML, XML Schema datatypes are used by providing an external link from the registry.

Table 2 provides a summary of how OWL language elements are mapped to ebXML class hierarchies. In this section, only some of these mappings are explained due to space limitations. A more complete treatment of the subject is given in [8].

In the implementation of the framework, we developed a tool to create an ebXML Classification Hierarchy from a given OWL ontology automatically by using the transformations described in this section. The OWL file is parsed [7], the classes together with their property and restrictions are identified, and the "SubmitObjectsRequest" is prepared automatically. This request is then sent to ebXML registry which in turn creates necessary classes and associations between them.

3. Service Discovery by Using OWL Semantics in ebXML Registries

```
<!DOCTYPE uridef[
 <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns">
 <!ENTITY tron "http://www.srdc.metu.edu.tr/2003/
          TravelOntology.owl">

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

<tron:ReserveAFlight rdf:ID="MyReserveAFlightService">
  <tron:departureFrom>  Ankara </tron:departureFrom>
  <tron:destinationTo>  Berlin </tron:destinationTo>
  <tron:price>  500  </tron:price>
  <tron:paymentMethod>  Credit Card  </tron:paymentMethod>
  <tron:minDiscountRate>  10  </tron:minDiscountRate>
</tron:ReserveAFlight>
</rdf:RDF>
```

Figure 6. "MyReserveAFlightService" Service Instance Description

Before we can start using semantics to discover services, there are a few more steps to be realized:

- A domain specific service ontology (like the one given in [14]) should be stored in ebXML registry by using the transformations described.
- To complete the picture, it is also necessary to define the semantics of individual service instances.
- Once the semantics of the service instance is described, it is necessary to relate it with the service advertised.
- Then the ebXML service registry can be queried to find the services according to their semantics. A nice feature of this querying is that it conforms to certain well defined query templates and therefore it can be parameterized as explained in Section 3.5.

3.1. Defining a Service Ontology

A service ontology provides the generic classes with properties and organize the concepts in a class hierarchy. Consider for example the travel ontology depicted in Figure 5, where the top level class "TravelService" inherits from OWL-S "Service" class. In this way, several properties of the "TravelService" class are conveniently defined by inheriting from the properties of the OWL-S "Service" class. Consider the "ReserveAFlight" service class given in Figure 5. A semantic description of this class might contain several common properties of such a service. For example, "paymentMethod" may be defined as a subproperty of OWL-S "ServiceProfile serviceParameter" property.

3.2. Defining Service Instance Semantics

Service instance semantics define the property values for an advertised service. As an example, Figure 6 shows "MyReserveAFlightService" described in OWL which is an instance of "ReserveAFlight" service class.

3.3. Relating Service Instance Semantics with the Services Advertised

Once the semantics of the service instance is described, it is necessary to relate it with the service advertised. The first step is to implement the Web service instances and to register their WSDL and OWL descriptions to the ebXML registry. Note that a service in ebXML is represented with a "Service" class in ebXML Registry. Both of these technical and semantic specification files (i.e., OWL and WSDL descriptions of the service instance) are stored in ebXML registry together with "Service" class that represents the service instance. Note that these files are stored external to the registry through 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. The OWL and WSDL descriptions of the service instance represent the semantic and technical specifications respectively and are accessed through the "SpecificationLink" instances.

Each service is published to an ebXML Registry via "SubmitObjectsRequest" with the corresponding WSDL and OWL files. The "SubmitObjectsRequest" for the example service instance given in Figure 6, is shown in Figure 7.

3.4. Relating the Generic Ontology with the Service Instance Semantics

In an ebXML compliant registry, *Classification* objects relate registry objects (which include Web services) with the nodes of a *ClassificationScheme*. That is, a *Classification* instance classifies a RegistryObject instance by referencing a node defined within a particular classification scheme [4].

As an example, assume that there is a service called "MyReserveAFlightService" (semantic description of this service instance is given in Figure 6) stored in the ebXML registry and a *ClassificationScheme* exits for the travel industry, namely, "TravelIndustry". This service is associated with the related Classification Schema through the "SubmitObjectsRequest". Figure 7 shows an example "SubmitObjectsRequest" which declares "ReserveAFlight" service classification node to classify "MyReserveAFlightService". Note that a registry object can be classified according to any number of classification schemes.

```
<rs:SubmitObjectsRequest
 <LeafRegistryObjectList>
  <!-- The web service definition -->
   <Service id="MyReserveAFlightService">
    <Name>
     <LocalizedString lang="en_US" value ="Reserve flight
      service"/>
    </Name>
    <ServiceBinding accessURI=
      "http://kadikoy.srdc.metu.edu.tr:8080/examples/
                                  ReserveFlightService">
      <SpecificationLink specificationObject="wsdl">
        <UsageDescription>
          <LocalizedString lang="en_US" value = "The
          service can be invoked by using Web Service
          Invocation Framework"/>
        </UsageDescription>
      </SpecificationLink>
      <SpecificationLink specificationObject="OWL">
        <UsageDescription>
          <LocalizedString lang="en_US" value =
          "The OWL file of the web service"/>
        </UsageDescription>
      </SpecificationLink>
    </ServiceBinding>
   </Service>
   <!-- The ExtrinsicObjects for the WSDL and OWL files.
Note that the actual files are repository items and are
attachments to the SOAP message carrying this request -->
    <ExtrinsicObject id="wsdl" mimeType="text/xml">
      <Name>
       <LocalizedString lang="en_US" value = "The WSDL
        document for the Reserve Flight Service"/>
      </Name>
    </ExtrinsicObject>
    <ExtrinsicObject id="OWL" mimeType="text/xml">
      <Name>
        <LocalizedString lang="en_US" value =
    "The OWL document for the Reserve Flight Service"/>
      </Name>
    </ExtrinsicObject>
    <Classification classificationNode="ReserveAFlight"
      classifiedObject="MyReserveAFlightService" />
  </LeafRegistryObjectList>
</rs:SubmitObjectsRequest>
```

Figure 7. "SubmitObjectsRequest" for the Example Service Instance of Figure 6

```
<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>
    <ServiceQuery>
      <ClassifiedByBranch>
        <ClassificationNodeQuery>
          <NameBranch>
            <LocalizedStringFilter>
              <Clause>
                <SimpleClause leftArgument = "value">
                  <StringClause stringPredicate = "Equal">
                    ReserveAFlightService</StringClause>
                </SimpleClause>
              </Clause>
            </LocalizedStringFilter>
          </NameBranch>
        </ClassificationNodeQuery>
      </ClassifiedByBranch>
    </ServiceQuery>
   </FilterQuery>
</AdhocQueryRequest>
```

Figure 8. An Example Query Retrieving the Instances of a Classification Node

3.5. Querying the Service Registry

Once the semantics is defined in this way, it is possible to provide automated access to service registry, through predefined query templates as described in the following:

- The first query template is used to obtain the properties of a generic class. All the properties inherited by the class through the "subClassOf" association are collected on the way. These properties can be depicted to the user through a GUI to obtain her preferred values.
- The second query template is used for locating service instances of a given generic class node in the class hierarchy. An example to the second query template that retrieves the services that are classified with the "ReserveAFlightService" is given in Figure 8. What is retrieved from the registry through this query template is a SOAP response containing the Service class which includes a collection of ServiceBinding UUID's. By using these UUID's, it is possible to access the UUID's of the OWL and WSDL files in the SpecificationLinks.
- The third query template is a content retrieval query to obtain the original OWL and WSDL files through the identifiers of the OWL and WSDL files in the SpecificationLinks.

The query templates are realized through ebXML Registry query facilities. The ebXML Registry supports two query capabilities : Filter Query and SQL Query [5]. 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.

4. Conclusions and Future Work

To be able to exploit service semantics, service registries should contain not only WSDL description of the services but also semantic description of the services instances.

This paper describes an engineering effort on how an ebXML compliant service registry can be augmented with an OWL service ontology and how service instance semantics can be used through standardized queries. There are two observations resulting from this experience:

- Ontologies can play two major roles in the Web services area: one is to provide a source of shared and precisely defined terms which can be used to dynamically discover, compose and monitor services. The other is to reason about the ontologies. When an ontology language like OWL is mapped to a class hierarchy like the one in ebXML, the first role can directly be achieved. However, for the second role, the reasoners to be developed for OWL will require OWL syntax and thus can not directly run on the ebXML class hierarchy. Yet the ontology may be reconstructed from the information in the class hierarchy.
- The only way to store semantic of individual service instances in ebXML registries is to store them external to the registry ("ExtrinsicObject" in ebXML terminology). And the content of external objects can not be queried through ebXML queries. If a mechanism is developed to store individual service semantics local to the registry, then the queries searching for services with required properties can be executed more efficiently.

References

[1] 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.

[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., Laleci, G., Kabak, Y., Cingil, I., "Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002.

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

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

[6] Horrocks, I., "DAML+OIL: a Description Logic for the Semantic Web", IEEE Data Engineering Bulletin, Vol. 25, No. 1, March 2002.

[7] Jena2 Semantic Web Toolkit, http://www.hpl.hp.com/semweb/jena2.htm

[8] Kabak, Y., "Exploiting Web Service Semantics through ebXML Registries", MS Thesis, Middle East Technical University, June 2003.

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

[10] Web Ontology Language (OWL) Reference Version 1.0, http://www.daml.org/2002/06/webont/owl-ref-proposed

[11] OWL-S, http://www.daml.org/services/daml-s/0.9/

[12] Paolucci, M., Kawamura, T., Payne, T., Sycara, K., "Importing the Semantic Web in UDDI", in Web Services, E-Business and Semantic Web Workshop, 2002.

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

[14] Wroe, C., Stevens, R., Goble, C., Roberts. A., Greenwood, M., "A Suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data", Intl. Journal of Cooperative Information Systems, to appear.

Building the Knowledge Economy

Issues, Applications, Case Studies

PART 1

Edited by
Paul Cunningham
Miriam Cunningham
and Peter Fatelnig

IOS Press

Ohmsha

P. Cunningham et al. (Eds.)
IOS Press, 2003

Exploiting Semantic of Web Services through ebXML Registries

Asuman DOGAC
Software Research and Development Center, Middle East Technical University, Inonu Bulvari, METU(ODTU) Campus, 06531, Ankara, Turkey
Tel:+90 312 2105598 Email: asuman@srdc.metu.edu.tr

Abstract: The aim of this tutorial is to present how Web service semantics can be exploited through ebXML registries. The tutorial starts with basic concepts including XML, Web services, Web service standards and builds on top of these concepts to address how to exploit service semantics in ebXML registries through simple but comprehensive examples. The aim is to make these topics easily digestible to the audience so that the companies can judge for themselves the possible benefits of these technologies.

1. Introduction

Since Extensible Markup Language (XML) is the common base of the technologies described in this tutorial, first a brief introduction to XML is presented. Over the recent years XML has become the "universal" standard for representing data. Starting out as a standard data exchange format for the Web, it has quickly become the fundamental instrument in the development of Web-based systems and standards.

1.1 XML in Brief

EDI	XML
.....	<PurchaseOrderRequest>
BGM+220+**1234ABCD**+9'	<PONumber>**1234ABCD** </PONumber>
DTM+137:**20030601**:102'	<PurchaseOrderDate>**20030601**</PurchaseOrderDate>
LIN+1'	<FirstLineItem>
PIA+5+**9344**:EN+1078341ITEM:VP	<ItemEAN_Identification no=**9344** />
QTY+21:**16**:EA'	<QuantityOrdered> **16** </QuantityOrdered>
PRI+AAA:**95**'	<UnitPrice> **95** </UnitPrice></FirstLineItem>
LIN+2'	<SecondLineItem>
...	...

Figure 1. Example Purchase Order Documents in EDI and XML

XML makes use of "tags" to give meaning and structure to data. Consider the XML example given in Figure 1. We give the structure of "PurchaseOrderRequest" document by defining its sub elements such as "PONumber" and "PurchaseOrderDate". With the tag "PONumber", we give a meaning to the string "1234ABCD", that this is a purchase order number. Similarly "PurchaseOrderDate" tag gives a meaning to the string "20030601" that it is a purchase order date. In Figure 1, we also provide the corresponding EDI message [7]. From this example, it is clear that:

- An XML document is a text document in contrast to a binary file which requires specialized software to process it.

- An XML document is human readable. That is, although it is processed through software, when necessary, a human can read it to make sense out of it.
- There are several public domain, open source tools to process XML documents such as editors and parsers.
- As long as the communicating parties agree on the tags, XML documents are machine processable. That is, if two parties agree on the structure and tags in the XML documents, they can write programs that will accept these XML documents as input and process them automatically.

1.2 Why XML is useful?

In order to explain why XML is useful, we compare some of the basic features of XML technology with that of EDI as shown in Figure 2.

XML	EDI
XML is an open human-readable, text format.	EDI documents are typically in a compressed, machine-only readable form (Please refer to Figure 1).
XML documents are typically sent via the Internet - i.e. a relatively low-cost public network.	EDI documents are typically sent via private and relatively expensive value-added networks (VANs).
XML has low flat-rate costs using existing Internet connections and relatively low-cost Web Servers.	EDI can involve high transaction based costs keeping up the connection to the EDI network and keeping the servers up and running.
XML has many low-cost and open source tools.	EDI was traditionally built in semi-isolation without being able to share resources with other programs.

Figure 2. A Comparison of XML technology with EDI

1.3 What is a Web service?

A Web service is a business function made available via Internet through XML artifacts by a service provider and accessible by clients that could be human users or software applications. Any application/component can be exposed as a Web Service. For example, one can have a Web service to accept purchase orders automatically.

The revolutionary aspect of Web services is that they provide interoperability at the interface level. This allows for clean integration across departments, organizations, and companies. The client who invokes the service and platform hosting the Web service can be different; they can be using different programming languages and operating systems. Note that one Web Service can make use of other Web services to perform a complex function.

Interactions among Web services involve three types of participants: service provider, service registry and service consumer as shown in Figure 3. Web services are stored in service registries. The universal standard for the technical specification of Web services is WSDL (Web Services Description Language) [15] and the standard for invoking Web services is SOAP (Simple Object Access Protocol) [11] which provides an XML based messaging and remote procedure call (RPC) mechanism.

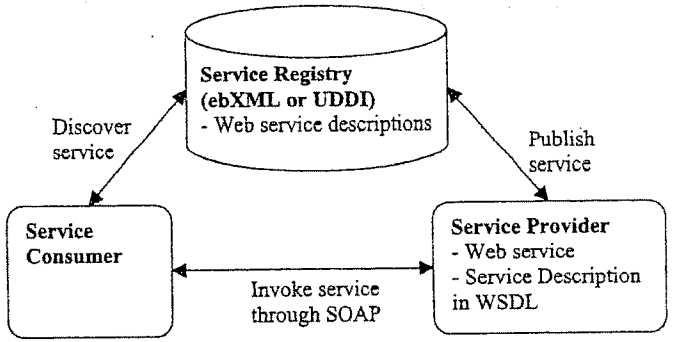


Figure 3. Web Service Model

There are two well known service registries: Electronic Business XML (ebXML) [6] Registries and the The Universal Description, Discovery, Integration framework (UDDI) [12] Registries. In this tutorial, we concentrate on ebXML.

Electronic Business XML is an initiative from OASIS and United Nations Centre for Trade Facilitation and Electronic Business [13]. 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. 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. Note that a Web service in ebXML is represented with a "Service" class in ebXML Registry.

1.4 Why do we need the semantics of Web services?

Web Service Description Language (WSDL) specifies only the technical interface of the Web services. WSDL provides the signature of the operations of the service, that is, the name, parameters and the types of parameters of the service. In other words, WSDL does not have any mechanism to express the semantic of the services like what the service is about or the meaning of its parameters.

On the other hand, Web services, like their real life counterparts, may have many properties such as:

- the methods of charging and payment,
- the channels by which the service is requested and provided,
- constraints on temporal and spatial availability,
- service quality, security, trust and rights attached to a service and many more.

As shown in Figure 4, to be able to describe these properties and later search for services according to their properties we need to describe the semantics of the service. This search needs to be done in a machine processable and interoperable manner. This in turn is possible only by describing the semantics of Web services through ontology languages. In other words, all the necessary properties of services can easily be defined through an ontology language and domain specific ontologies can be developed by standard bodies. It

is a good idea to ground domain specific ontologies in upper ontologies since in this way they are more consistent and it becomes easier to integrate them within distributed heterogeneous systems.

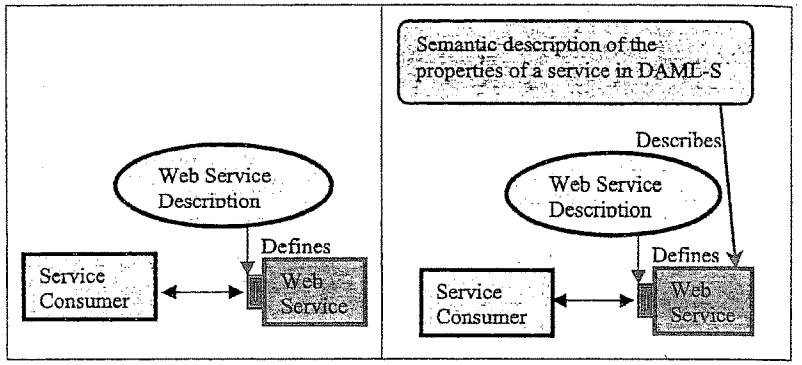


Figure 4. WSDL defines the interface of the service; semantic describes service itself

Currently, describing the semantic of Web in general [1], and semantic of Web services in particular are very active research areas. World Wide Web Consortium has started the initiative to develop Semantic Web and a semantic markup language for publishing and sharing ontologies, namely Web Ontology Language (OWL) [14], is being developed for this purpose. OWL is derived from DAML+OIL [2] by incorporating learnings from the design and application use of DAML+OIL. It builds upon the Resource Description Framework [9, 10]. It should be noted that there are only minor differences between OWL and DAML+OIL.

There are a number of efforts for describing the semantics of Web services. Among these DAML-S [3] 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.

2. Describing the Semantics of Web Services, DAML-S

DAML-S defines an upper ontology for defining the semantics of Web services. It is based on DAML+OIL and aims to enable the automation of the following functionalities [3]:

- Web service discovery: Web service discovery involves the automatic location of Web services that provide a particular service and that adhere to requested constraints.
- Web service invocation: Web service invocation involves the automatic execution of an identified Web service by a computer program or a software agent.
- Web service composition and interoperation: This task involves the automatic selection, composition and interoperation of Web services to perform some tasks, given a high-level description of an objective.
- Web service execution monitoring: Individual services and, even more, compositions of services, will often require some time to execute completely. Users may want to know during this period what the status of their request is.

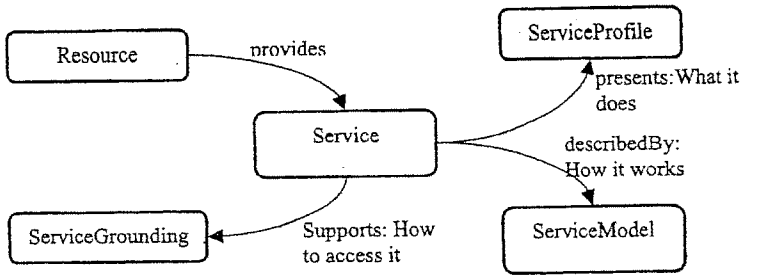


Figure 5. Describing the upper ontology of services

The top level class in DAML-S service taxonomy is the "Service" class as shown in Figure 5. Service class has the following three properties:

1. 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.
2. describedBy: The range of this property is *ServiceModel* class. That is, the class Service is *describedBy* a *ServiceModel* to specify how it works.
3. supports: The range of this property is *ServiceGrounding*. That is, the class Service *supports* a *ServiceGrounding* to specify how it is used.

2.1 How do we define and use service semantics?

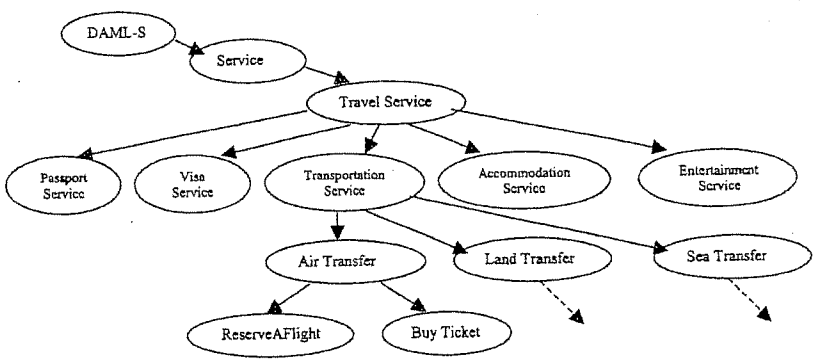


Figure 6 An example classification hierarchy for travel industry

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). UDDI does not provide an internal mechanism to store generic service semantics. ebXML, on the other hand, through its classification hierarchy mechanism allows domain specific ontologies to be stored in the registries.

An ebXML 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.

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.

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* [4,5].

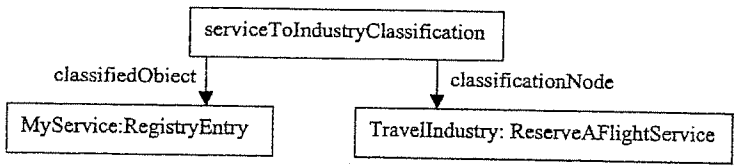


Figure 7 Relating service instances with the nodes in the classification hierarchy

Consider for instance the classification hierarchy example given in Figure 6 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 advertised 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 as shown in Figure 7, 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" as shown in Figure 8, "MyService" also inherits these properties as shown in Figure 9.

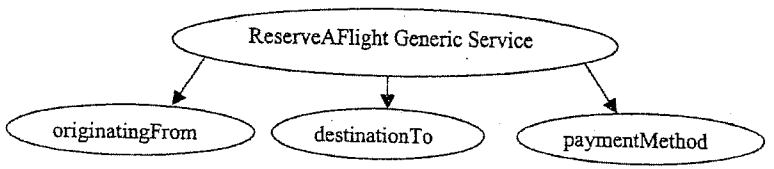


Figure 8. Properties of "ReserveAFlight" Generic Service

```
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY tron "http://www.srdc.metu.edu.tr/2003/TravelOntology.owl">
<rdf:RDF
   xmlns:rdf =    "&rdf;#"
   xmlns:tron =   "&tron;#"
<tron:ReserveAFlight rdf:ID="MyService">
 <tron:departureFrom>  Ankara </tron:departureFrom>
 <tron:destinationTo>  Bologna </tron:destinationTo>
 <tron:price>  500  </tron:price>
 <tron:paymentMethod>  Credit Card  </tron:paymentMethod>
 </tron:ReserveAFlight>
</rdf:RDF>
```

Figure 9. An Example Service Semantics

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 [5]: 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".

2.2 What do we gain from service semantics?

When services are described according to a service ontology, it becomes possible to discover and compose services automatically. Consider for example the service definition given in Figure 9, where service properties are described in OWL by conforming to travel service ontology of Figure 6:

1. First, the service description is both human readable and machine processable. That is, one can write a program to find and process such services dynamically.
2. Secondly there are very many public tools such as APIs and reasoners to process such descriptions in DAML and RDF. OWL tools are expected to appear shortly. In other words, by conforming to a standard ontology language while describing semantics allows us to use publicly available tools.
3. Conforming to standards solves the interoperability problem; an organization complying with these standards can easily exchange machine processable information with other organizations.
4. The service name "MyService" does not convey information about what the service is about. However by making it an explicit member of the "ReserveAFlight" node in the ontology, we give it a concensually agreed meaning that it is a flight reservation service. Hence, when a program is searching the service registry, it is able to identify this service as a flight reservation service.
5. Furthermore, we were able to describe the properties of this service which can be used in service discovery. For example, when an agent (human or software) searches a flight reservation service that accepts "credit card payment", this service can easily be located since its "paymentMethod" contains this information.

2.3 What is the role of ebXML registry in supporting the semantics of Web services?

ebXML registry helps to support the semantic with the following mechanisms:

1. It is possible to define the properties of registry entries through "slots" which gives way to define the properties of Web services;
2. It is possible to store classification hierarchies in the services registry. Furthermore, ebXML allows relating the services advertised in the registry with the semantics defined in the classification hierarchy through explicitly declared classification objects. This makes it possible to query the registry to find out services according to their semantics.

3. Conclusions and Recommendations

Two trading partners can communicate and do transactions with each other automatically as long as they comply with the same B2B standard. That is, conforming to a B2B standard, such as ebXML, isolates a company's back-end system from the external world. Exploiting the semantics of data on top of conforming to a standard B2B framework brings further benefits in automating the business processes.

However there are very many B2B standards some of which is overlapping; some complements one another and some competing with each other. It should be noted that unless all trading parties conform to the same standard there is still a need to develop interoperability mechanisms among these standards. The topics briefly mentioned over here due to space limitations, such as other B2B frameworks, will be elaborated during the tutorial presentation.

Finally, the authors wish to gratefully acknowledge the Scientific and Technical Research Council of Turkey for supporting this work through Project No: EEEAG 102E035.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.
[2] DAML+OIL, http://www.w3.org/2001/10/daml+oil
[3] 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.
[4] ebRIM: ebXML Registry Information Model v2.1, June 2002, http://www.ebxml.org/ specs/ebRIM.pdf.
[5] ebRSS: ebXML Registry Services Specification v2.1, June 2002, http://www.ebxml. org /specs/ebiRS.pdf.
[6] ebXML, http://www.ebxml.org/
[7] Electronic Data Interchange, http://www.unece.org/cefact/index.htm.
[8] OASIS ebXML Registry Reference Implementation Project (ebxmlrr), http://ebxmlrr. sourceforge.net/
[9] RDF Schema: Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.
[10] RDF Syntax: Resource Description Framework Model and Syntax Specification, W3C Recommendation, 1999, http://www.w3.org/TR/REC-rdf-syntax.
[11] SOAP: Simple Object Access Protocol, http://www.w3.org/TR/SOAP/
[12] UDDI: Universal Description, Discovery and Integration, http://www.uddi.org
[13]JUN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/
[14] Web Ontology Language (OWL) Reference Version 1.0, http://www.daml.org/2002 /06/webont/owl-ref-proposed
[15] WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl

P. Cunningham et al. (Eds.)
IOS Press, 2003

Experiences in ebXML-based Interoperability

Flavio BONFATTI[1], Paola MONARI[2], Emanuela MONTANARI[2]
[1]*University of Modena and Reggio Emilia, Via Vignolese 905 – 41100 Modena, Italy*
Tel: +39 059 2056138; Fax: +39 059 2056129; Email: bonfatti@unimo.it
[2]*SATA srl ,via Notari 103 – 41100 Modena, Italy*
Tel: +39 059 343299, Fax +39 059 343299; Email: p.monari@satanet.it

Abstract: The paper describes three experiences of use of ebXML in SME oriented IST projects, devoted to (i) set-up and experiment tools for aggregating the offer of different SMEs acting in the bio-medical field to answer large calls for tender, (ii) provide innovative modelling, planning and monitoring tools for the extended enterprise, (iii) put clusters of individual truck owners in condition to interact timely and properly. The last two experiences already constitute a step forward in the Transport Unified Protocol (TUP) definition and adoption, although the TUP scope is much broader, as it is particularly oriented to support multi-modal and cross-boundaries transportation. Finally, lessons learnt from the gained practice are presented.

1. Introduction

Manufacturing SMEs are the backbone of the European economy, and are more and more oriented to act on a worldwide basis. Since many years they are used to export their products to other countries, and recently the pressure towards globalisation pushed them to de-localise their supply chain to foreign regions where they find favourable conditions for cost reduction (lower labour costs, lower taxes and so on).

This means that the incidence of the transportation costs on the overall company budget is significantly increasing, and this implies also a relevant amount of clerical work to plan and monitor transportation missions. Such clerical work is so heavy that the majority of SMEs try to delegate the management of internal and external logistics, at least to some extent, to other actors: specialised logistics operators, customers and suppliers. This delegation typically results in a lack of control of transportation time and costs, with problems in ensuring the due service level to the company partners and in general no possibility to optimise such a critical value chain phase.

On the other hand, SMEs are approaching e-commerce much slowly than expected. In most cases they are willing to exchange data with their counterparts via ICT means, but they still lack proper software applications that allow such smooth interaction, and they still maintain doubts on security and legal issues. These problems are critical for SMEs, but not completely solved also for large manufacturing and service company all over the world. The mission of a recent initiative of the DG-INFSO, the Single European Electronic Market (SEEM [1]), is to speed up the process of studying and realising a new generation of software tools that put any company in condition to interact with anyone else. This calls from contributions of experts from technology, regulations, standardisation, as well as representatives of enterprises, at the European and worldwide level.

The Transport Unified Protocol (TUP) for the standardisation of goods trip and accompanying documents is compliant with the SEEM ideas, and it is particularly important as it deals with an area, logistics management, that traditionally lacks effective and standardised ICT solutions. An essential part of the SEEM, very important also for the

Building the Knowledge Economy

Issues, Applications, Case Studies

PART 2

Edited by
Paul Cunningham
Miriam Cunningham
and Peter Fatelnig

IOS Press
Ohmsha

Building the Knowledge Economy: Issues, Applications, Case Studies
P. Cunningham et al. (Eds.)
IOS Press, 2003

Context Framework for Ambient Intelligence

Asuman DOGAC, Gokce B. LALECI, Yildiray KABAK
Software Research and Development Center, Middle East Technical University,
Inonu Bulvari, METU(ODTU) Campus, 06531, Ankara, Turkey
Tel:+90 312 2105598 Email: asuman@srdc.metu.edu.tr

Abstract: ISTAG has developed the concept of Ambient Intelligence (AmI) to provide a vision on how the Information Society will develop. The aim is to achieve better integration of technology into our environment, so that people can freely and interactively use it. An indispensable component of AmI is user context. We claim that to realize the vision of AmI the following issues need to be addressed: the user context information need to be interoperable and machine processable since it will be exploited by a variety of devices. Therefore it is necessary to develop context ontologies. Furthermore, the user context should be globally accessible; this necessitates developing context servers. Considering that AmI devices accept input in different mark up languages; the context server needs to recognize the device and provide the information in the format that can be accepted by the device. More importantly, user context information should only be provided to the authorized entities and the user should be able to specify how much of that information should be available to whom. Finally, AmI is not restricted to local context obtained through sensors; integrating this information with the user profile and preferences opens up a way for real life applications where the user context can be exploited for Web service discovery and composition.

In this paper, we present a framework to address exactly these issues. We show how context ontologies can be developed, stored, queried. We then describe mechanisms for the privacy and security of user context. Finally we present the way to exploit user context for Web service discovery and composition through ebXML registries.

1. Introduction

Ambient Intelligence (AmI) stems from the convergence of three key technologies: Ubiquitous Computing, Ubiquitous Communication, and Intelligent User Friendly Interfaces [6]. The increased availability of commercial, off-the-shelf sensing technologies and the prevalence of powerful, networked computers and devices are bringing this vision much closer. However there are issues to be addressed before this vision can be realized. One of these issues is handling user context information.

We define user context to be any information that can be used to characterize the user and her situation. The user context can thus include a wide variety of information coming through sensors such as the current temporal and spatial location, and the stored information such as user preferences, and profile. The current context-aware applications are usually build in an ad hoc manner and lack the generality and standards to be of use in AmI environments.

We claim that to realize the vision of AmI the following issues need to be addressed:

- Developing Context Ontologies: The context should be machine processable and interoperable since it will be accessed by a variety of devices. A solution is to develop ontologies for user context. In other words, to make user context information accessible by agents (software or human), several modular ontologies need to be developed addressing different parts of user context such as temporal aspects, spatial aspects, profiles, etc. Parts of already existing ontologies may be used for this purpose.

- Developing mechanisms to respect user's right to privacy when revealing user context information: User context information should only be provided to the authorized entities and the user should be able to specify how much of that information should be available to whom. For example the user may wish that certain aspects of her context should only be available to certain types of agents.
- Developing globally accessible Context Servers: The context information should be available to any authorized agent at any time, any where in a secure manner: This necessitates developing globally accessible, secure "context servers", that is, the user context information should be available any where, any time to a variety of devices from desktops to mobile devices. Since these devices accept input in different mark up languages; the context server needs to recognize the device and provide the information in the format that can be accepted by the device. Note that if the user permits, information on user activities should be collected to further improve user context.
- Integrating context-awareness with Web service discovery: AmI is not restricted to local context obtained through sensors; integrating this information with the user profile and preferences opens up the way for real life applications where the user context can be exploited for Web service discovery and composition.

To demonstrate our case, we present a part of one of the scenarios given in [6]:

"After a long haul flight Maria lands to an airport in a Far Eastern country. The immigration officer in this country has been replaced by a device. The immigration device, through its sensor, detects the identities of the people entering to the immigration area and performs visa and passport control. Maria carries a P-Com device that has her identity information and her personal software agent has arranged for her visa. Therefore she is able to stroll through the immigration. There is a rented car waiting for her at the exit and her hotel has been reserved thanks to her personal software agent."

We will use this scenario to describe the context information requirements needed by AmI environments: When Maria arrives at the immigration hall, the immigration device, through its sensor, can detect the identity of Maria from her P-Com. There are several issues to be considered here:

1. The identity information should be understandable by any authorized device any where which implies there has to be standards in this respect.
2. Through the identity of the person the necessary information should be accessible. In the scenario presented, this is the passport and visa information of Maria. This information can be obtained by the immigration device either by querying the context server or if the P-Com also has the information, it can transmit it. However in any case this information should be machine processable (the immigration device has to process it without human intervention) and interoperable (this data originating from Europe, should be processable in the Far East). In other words, ontologies need to be developed to make user context automatically accessible by agents.
3. Due to the personal and proprietary nature of the context information, access must be limited to authorized entities and the user should be able to state how much information to disclose and to whom. In the above scenario, the immigration device should get only the location, visa and passport part of the user context. Indeed if context information can not be disclosed on a role or personal identity basis, if it can not be securely transmitted over untrusted networks and if it is not possible to authenticate communications with remote hosts, this will destroy many of the promised rewards of the Ambient Intelligence.
4. To provide security and global accessibility, the stored part of the context information (such as user profile and preferences) should be available through "trusted context servers". To provide for interoperability context servers to be developed must be based

on open standards. Note that there could be more than one trusted context server on the Web to provide for scalability.

5. Furthermore, since the user context information should be available to any device from desktops to mobile devices and these devices accept input in different mark up languages; the context server needs to recognize the device type and provide the information in the format that can be accepted by the device.
6. Finally, it should be possible to exploit context to discover and compose Web services for a user. For example, in the above scenario, it is Maria's software agent which arranged the details for her trip. To realize this, the agent needs to discover services based on her context. For example, to obtain a visa for her, the agent might get her itinerary from her context; invoke a service at the related embassy by providing the information necessary to get a visa. This information may include her nationality, her previous trips to the country, etc. which can again be obtained from the context server.

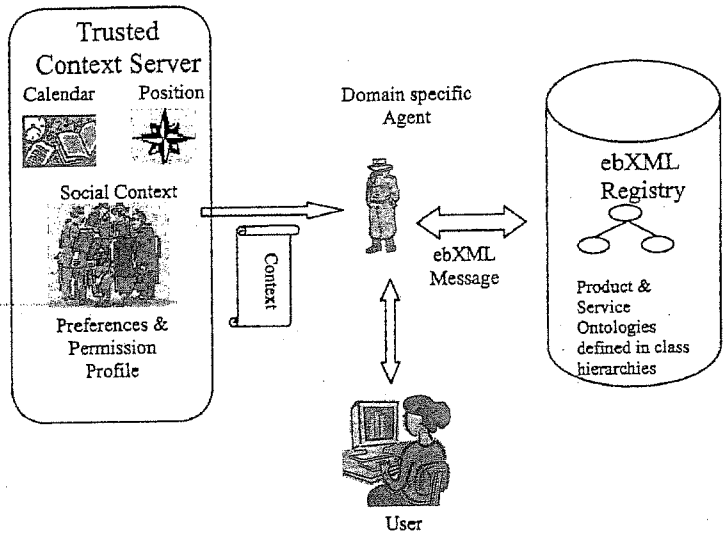


Figure 1. Overall Architecture of the System

In this paper we describe a framework to address these issues. Figure 1 shows the overall architecture of the system. User context is stored in an ontology server. A view based mechanism is used to provide for the privacy and security of user context information. Services are stored in a domain specific ebXML registry and a JADE agent is associated with the registry. When a user needs a Web service, the agent queries the service registry to find the explicit members of the given ontology class.

2. Describing User Context Through an Ontology Language

There is a need to develop context ontologies to realize the vision of AmI. A user context can involve, identity information, profile (e.g., the expertise area, company information), preferences (e.g., entertainment preferences, travel preferences), spatial information (e.g., location, orientation, speed and acceleration), temporal information (e.g., time of the day, date, and season of the year), environmental information (e.g., temperature, air quality, and light and noise level), social situation (e.g., the people nearby), resources that are nearby (e.g., accessible devices, sensors), availability of resources (e.g., battery, display, network, and bandwidth), physiological measurements (e.g., blood pressure, heart rate, respiration rate, muscle activity), activity (e.g., walking, talking, running), schedules and agenda

setting. As mentioned previously, some of this information may be coming from the sensors; some of it are stored in context servers.

Ontologies provide machine-processable semantics of information sources that can be communicated between agents (software and human). The following properties of ontologies make them an indispensable tool for defining semantics:

- They have a formal specification. This property makes them machine processable. For example they can be queried through query languages to obtain the desired information.
- Ontologies define shared conceptualizations, that is, an ontology captures consensual knowledge meaning that it is not restricted to an individual but accepted by a group.

In practical terms, developing a context ontology includes determining the information relevant to context and then:

- Defining classes in the ontology
- Arranging the classes in a taxonomic (subclass, superclass) hierarchy,
- Defining properties (i.e. slots) and describing allowed values (facets) for these properties,
- Filling in the values for properties for instances,
- Defining the relations among the classes.

In order to enable as much reuse as possible, ontologies should be developed in small modules. If well-designed modular ontologies can be developed, new ontologies can be constructed by assembling these modules.

There are several ontology languages. Among these Web Ontology Language (OWL) [10] is a semantic markup language being developed by the World Wide Web Consortium for publishing and sharing ontologies. OWL is derived from DAML+OIL [4] by incorporating learnings from the design and application use of DAML+OIL. It builds upon the Resource Description Framework [11,12].

Developing domain specific ontologies are the responsibility of standard bodies and industrial consortiums. It should also be noted that there are libraries of reusable ontologies on the Web and in the literature such as the Ontolingua ontology library, or the DAML ontology library.

3. Storing Context Ontologies and Security and Privacy Issues

Ontologies are stored in knowledge-bases. Several tools have been developed for this purpose. However where and how to store the context ontologies depends also on the level of security and privacy the repository can provide.

There are two aspects of security and privacy in the context platform proposed:

- Security and privacy of data coming from the sensor devices. In our example, this corresponds to the signals coming from Maria's P-Com device and this should only be accessible to the authorized devices such as the immigration authority device. There are a number of systems developed for this purpose. For example, in Solar System [8], such an access-control mechanism is described where sensor data is represented as events flowing from sources through operators (which filter, transform, or aggregate events into new events) to applications. Each event is tagged with an access control list (ACL) thus restricting the receipents.
- Security and privacy of data obtained from the context server. Due to the personal and proprietary nature of the context information, access to the context server must be limited by authorized entities and the user should be able to state how much information to disclose and to whom. For this purpose, a mechanism similar to the well-established view mechanism for relational and object databases, is necessary. In [13], a view mechanism, implemented as a part of KAON [7] server, is described. As the query

language of the view mechanism, RQL [1] is used. The main concerns of the authors in this work are as follows:

- The view should again be based on an ontology
- The approach should be functional, that is, it should allow to create views based on other views.

To achieve these goals, they differentiate between views on classes and views on properties and view definitions are restricted to queries which return either views on classes or views on properties.

Such a view mechanism can be used for providing the security and privacy of context information in a context server. A user can define how much of her context information should be available to whom by defining views on her context and granting different access rights to different agents (human or software) on these views. Consider for example the context hierarchy given in Figure 2. It is possible to define views on the user context and grant access rights to different users as shown in the following example:

```
CREATE CLASS VIEW ScheduleOfMaria
        SUBCLASSOF Schedule
        SELECT X
        FROM ContextServer
        WHERE X.Name= "Maria"
GRANT SELECT ON ScheduleOfMaria TO PersonalAgentOfMaria
```

In this example, a view is created on the "schedule" class is created which contains the schedule information of the user "Maria" and this information is granted to her personal agent.

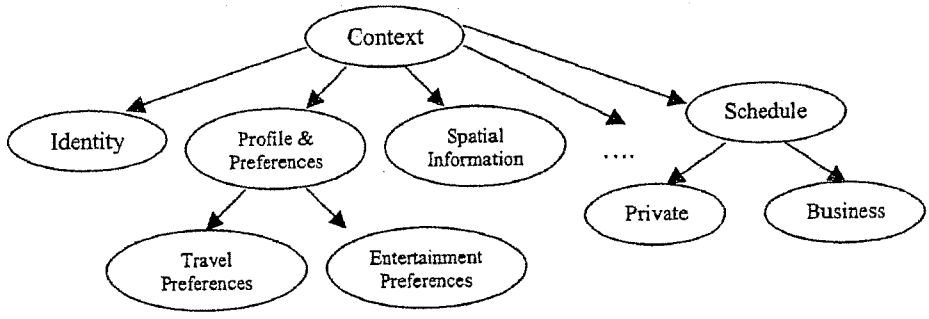


Figure 2. An example Context hierarchy

4. Exploiting User Context for Web Service Discovery

In order for agents to select services in a context–sensitive manner, they must be able to discover services based on their semantic description. A key issue in semantic discovery of services is to relate the service advertised in the registry with its semantic description. ebXML registry allows metadata to be stored in the registry through a "Classification" mechanism which helps to classify the objects (i.e. services) in the registry. In the following we describe how user context can be used to discover services.

Consider our running example. When Maria's agent discovers that she will be travelling to a country at the Far East, it starts to arrange her trip. The personal agent first obtains Maria's schedule by querying the context ontology given in Figure 2 as described in Section 3. If Maria does not have a valid passport for her trip, it is necessary to obtain a passport for her. Thanks to the e-government initiatives through out Europe, we can expect

the passport services to be electronically available not in a distant future. Then the personal agent of Maria can invoke this Web service of her government to obtain her a passport. The information needed for this purpose can be obtained from her context information.

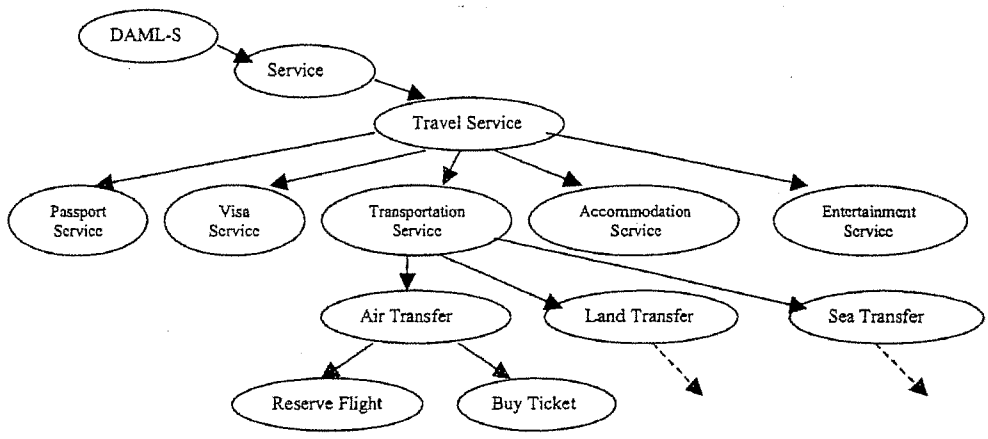


Figure 3. An example Travel Ontology

The personal agent then checks whether she needs a visa for the country she will be travelling and again it finds the service of the related Embassy on the Web to get a visa.

To further plan for the travel, the agent contacts the domain specific registry agent of a service registry for the "Travel" domain. Consider, for example, the "Travel" service ontology given in Figure 3 which is a classification schema that defines the semantics of a travel services. The domain specific registry agent queries the service ontology stored in the ebXML registry by predefined query templates to find the subclasses of the "Travel Service". This domain specific agent also has rules that are provided by a domain expert. These rules state the possible service ontology classes that might follow a given service ontology class. For example, a rule may state that a transportation service will follow a visa service. Using such rules the agent may infer that the next step is arranging a Transpostation Service. Again consulting to the ebXML classification Schema the agent discovers that there are several alternatives for transportation like, air, land and sea. Here the domain specific registry agent contacts with Maria's personal agent which in turn queries Maria's context to find out her "Travel Preferences".

In this way, the agent finds a generic service description (a node in the class hierarchy) from the ebXML classification hierarchy that is suitable for the user. For example, if Maria's travel preferences are air travel, her personal agent will convey this information to the domain specific registry agent so that the agent can access the instances of this specific generic service to find out the needed service. The details of this process are as follows: The services and the service ontology objects in the classification hierarchy are related with Classification objects. The agent retrieves the instances of the "Reserve A Flight" generic services by predefined query templates. The service instances in the registry have some slots for defining the properties of it. For example the instances of the "Reserve a Flight" generic services have a slot for relating them for specific Airline companies. The personal agent checks the profile of the user find out that Maria has a bonus card from a specific Airline company and chooses the corresponding airline service among the existing instances of the "Air Transfer" services registered to the ebXML registry.

Then a hotel may be reserved for Maria in the same way, by checking her preferences from the context server, and querying the ebXML registry.

After arranging the transportation and accomadation for Maria, the domain specific registry agent checks the Travel Ontology registered in the ebXML registry, and realizes that it can arrange an entertainment service for Maria. Then the personal agent querries the schedule of the user from the context server and finds out that she has a spare time between her meetings, and she likes to watch science fiction movies. The domain specific registry agent retrieves the insances of services selling cinema tickets, querries their timetables and offers Maria the possible choices.

With the help of her personal agent, Maria has a full travel plan before she arrives the Far East country. When she arrives at the airport, since her visa and passport have been arranged by her personal agent, she passes through the immigration office without stopping.

5. Related Work

The current context-aware applications are usually build in an ad hoc manner and lack the generality and standards to be of use in AmI environments.

Authentication services like Microsoft Passport [9] and AOL Screen Name [2], store and manage personal user data and provide single sign-in identity in different sites and pass personal information more easily. However the stored personal data is generally limited to user identification and user contact information that can be used in basic e-commerce sessions.

[5] describes the ePerson project developed at the HP Labs. An ePerson is a personal representative on the net that is trusted by a user to store and make available personal information under appropriate controls. Such personal information includes user profiles, shared content and shared meta-data (such as annotations, comments, ratings and categorisations). However the details of this project is not available in the literature.

Among several approaches for privacy management using service policies and privacy preferences, the most mature one is the Platform for Privacy Preferences Project (P3P) [3] developed by the World Wide Web Consortium (W3C). P3P enables Web sites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents like Web browsers. It should be noted that P3P is for Web sites and does not use Web semantics.

6. Conclusions and Recommendations

In this paper we describe a framework for handling context information in Ambient Intelligence environments. For this purpose, we describe the need to develop context ontologies for describing user contexts; the mechanisms necessary to provide for the security and privacy of context information. Finally, we explain how user context information can be exploited to find and compose semantically enriched services.

The aim of this paper is to describe that there are a considerable number of promising technologies and initiatives such as off-the-shelf sensing technologies, ubiquitous computer networks, ontology servers, agent technologies for mobile devices, Web service registries, Web standards related with security and privacy and emerging standards for the Web semantics, which can be exploited in developing AmI environments. Further research and development activities in almost all of these areas are being addressed in FP6 IST program. What is needed now is to work on frameworks where these technologies can be adequately combined, deployed and used in real production environments. It should be noted that AmI applications require cross-disciplinary and cross-sectoral capabilities and the actors involved should include not only the industry but also the governments and more importantly the people since to be acceptable AmI needs to be driven by human needs, not technologically determined ones.

There are also innumerous business opportunities. To mention but a few, consider our running example: The airport with the immigration device saves a lot of time for the

passengers. Security checks are facilitated. E-government services mentioned like "passport" and "visa" also save time and facilitate the life of people helping to create the vision of information society with more efficient services support, and user-empowerment.

Further possible business models are identified in [6] for the AmI business landscape:

- Initial premium value niche markets in industrial, commercial or public applications. For instance, in our running example this corresponds to developing an "immigration officer device" with it corresponding software. Yet another opportunity is designing and developing a P-Com device which can be considered as the next generation of current mobile devices.
- Start-up and spin-off opportunities from identifying potential service requirements and putting the services together that meet these new needs. For example developing semantically enhanced interfaces to Web based applications and services to make them machine processable may create further opportunities. In Maria's case this corresponds to semantically enhanced e-government services like "visa" and "passport" which are accessible to her personal agent without human intervention.

Finally the authors wish to gratefully acknowledge the Scientific and Technical Research Council of Turkey for supporting this work through Project No: EEEAG 102E035.

References

[1] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, Proc. of the Second International Workshop on the Semantic Web (SemWeb2001), May 2001.
[2] AOL Screen Name, http://my.screenname.aol.com
[3] Cranor L., Langheinrich M., Marchiori M., Presler-Marshall M., Reagle J., "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification", W3C Recommendation, http://www.w3.org/TR/P3P, April 16, 2002.
[4] DAML+OIL, http://www.w3.org/2001/10/daml+oil
[5] e-person: Personal Information Infrastructure, http://www.hpl.hp.com/semweb/e-person.htm
[6] ISTAG Scenarios for Ambient Intelligence in 2010, http://www.cordis.lu/ist/istag.htm
[7] Karlsruhe Ontology (KAON) tool suite, http://kaon.semanticweb.org/.
[8] Minami, K., Kotz, D., "Controlling Access to Pervasive Information in the "Solar System", Technical Report, TR2002-422, Dartmouth University, Feb. 2002, http://www.cs. dartmouth.edu/reports/abstracts /TR2002-422/
[9] Microsoft Passport, http://www.microsoft.com/myservices/passport.
[10] Web Ontology Language (OWL) Reference Version 1.0, http://www.daml.org/2002 /06/webont/owl-ref-proposed
[11] Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.
[12] Resource Description Framework Model and Syntax Specification, W3C Recommendation, 1999, http://www.w3.org/TR/REC-rdf-syntax.
[13] Volz, R., Madche, A., Oberle, D., "Towards Views in Semantic Web", 2nd Intl. Workshop on Databases, Documents and Information Fusion (DBFUSION 02), July 2002, Karlsruhe, Germany.