2005_219

TÜBİTAK

TÜRKİYE BİLİMSEL VE TEKNOLOJİK ARAŞTIRMA KURUMU

THE SCIENTIFIC AND TECHNOLOGICAL RESEARCH COUNCIL OF TURKEY

SAĞLIKLI BİLGİ SİSTEMLERİ İÇİN WEB SERVİSLERİ TABANLI BİR P2P BİRLİKTE İŞLERLİK PLATFORMU

(A WEB SERVICE BASED P2P INTEROPERABILITY PLATFORM FOR MEDICAL INFORMATION SYSTEMS)

70125 37

PROJE NO : 104E013 1-30, Ekler

Elektrik, Elektronik ve Enformatik Araştırma Grubu

Electric, Electronics and Informatics Research Grant Committee

SAĞLIKLI BİLGİ SİSTEMLERİ İÇİN WEB SERVİSLERİ TABANLI BİR P2P BİRLİKTE İŞLERLİK PLATFORMU

(A WEB SERVICE BASED P2P INTEROPERABILITY PLATFORM FOR MEDICAL INFORMATION SYSTEMS)

70125 37

PROJE NO : 104E013 1- 30, Ekler

PROF.DR. ASUMAN DOĞAÇ

MAYIS 2005
ANKARA

ÖNSÖZ

Bu projede sağlık kurumlarının birlikte işlerliğini sağlamak amacıyla bir altyapı geliştirilmiştir. Bu amaçla, sağlık kurumları kendi ontolojilerini varolan ontolojileri baz alarak tanımlamakta ve hizmetlerini anlamsal olarak zenginleştirilmiş Web Servisler olarak diğer kurumlara açmaktadır. Kullanılan P2P altyapısı ve bu yapıya Web Servis depolarını yerleştirerek Web Servisler ister tek bir kullanıcı da ister P2P ortamda herhangi bir Web Servis deposunda olsun kolayca bulunmaktadır.

Sağlık kurumları farklı standartları baz alabildiği için, kurumlararası birlikte işlerliği aracı bir bileşen sağlamaktadır. Bu aracı bileşen iletişim kurmaya çalışan iki tarafın ontolojilerini inceleyip arada iletilen mesajların ontoloji eşlemesiyle tarafların anlayabileceği şekle çevirmektedir. Sağlık personelinin sürekli olarak mobil olmasından dolayı geliştirilen altyapıya taşınabilir araçlarla da erişebilmek amacıyla arayüzler geliştirilmiştir.

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.

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

İÇİNDEKİLER

TABLO ve ŞEKİL LİSTESİ

ÖZ

Günümüzde kullanılan Sağlık Bilgi sistemlerinin büyük çoğunluğu özel kullanımlar için geliştirilmiş olup çoğunlukla sağlık kurumlarının belli bölümlerine hizmet vermektedirler. Bunun ötesinde bir hastanın sağlık kayıtları birlikte çalışamayan birden fazla sağlık kurumunda bulunuyor olması mümkündür. Tüm bunlar sağlık personelinin bir hastanın sağlık kayıtlarının tamamına ulaşmasını çok zorlaştırmaktadır. Bu projenin amacı bu problemlere çözüm olarak, Sağlık Bilgi Sistemleri için Web Servis tabanlı P2P altyapısı üzerinde bir "birlikte işlerlik" platformu oluşturmaktır.

Günümüzde yaygın olarak kullanılan sağlık standartlarından, CEN TC251 (CEN TC251), GEHR (GEHR) ve HL7 (HL7) da sağlık bilgi sistemlerininin birlikte işlerlik problemini çözmeyi amaçlamaktadır. Ancak bu standartlara uygun geliştirilmiş çok sayıda sistem olmasına rağmen, sağlık sektöründe makinalar arası birlikte işlerlik problemi aşagıda sıralanan sebeplerden dolayı hala çözülebilmiş değildir:

- Öncelikle, adı geçen standartlardan bazıları tamamen otamatik olarak veri işlemesini destekleyecek düzeyde birlikte işlerliği hedeflememektedir. Örneğin, Elektronik Hasta Kaydı tabanlı standartlardan ENV 13606 (CEN) ve GEHR hasta kayıtlarının bilgisayar yardımıyla daha iyi anlaşılabilmesi ve daha iyi kategorize edilebilmesi amacıyla; hasta kayıtlarının anlamlı parçalarını tanımlamışlardır.
- Sistemler standartlardan birine uygun geliştirilmiş olsalarda başka bir standarda uyan bir sistemle birlikte işlerlik problemi çözülememektedir. Aslında bütünleşik sağlık bilgi sistemlerinin önündeki en büyük engel degişik standartların birbirleriyle iletişim problemidir.

Tüm bunların yanında, Web Servis Modeli sağlık endüstrisine aşılması zor birlikte işlerlik problemlerini çözmek icin ideal bir platform sağlamaktadır. Web Servisler varolan yazılımları sarmalayarak farklı uygulamaların birlikte işlerliğini sağlayabilirler. Sağlık alanında Web Servis kullanımının getirileri şu şekilde özetlenebilir:

- Elektronik hasta kayıtlarının dokümantasyonunu standartlaştırmak yerine kayıtlara WSDL (WSDL) ve SOAP (SOAP) yolu ile ulaşılmasını standartlaştırarak sağlık bilgi sistemlerinin birlikte işlerliğini mümkün kılarlar.
- Sağlık Bilgi Sistemleri aynı veriyi değişik şekillerde sunmak için oluşturulan standartların hızla çoğalması problemi ile karşı karşıyadır. Web Servisler birbirinden farklı hatta birbirlerine rakip standartlara uyumlu yazılmış uygulamaların pürüzsüz bir şekilde birlikte işlerliğini mümkün kılar.
- Web Servisler sağlık kuruluşlarının varolan servislerininin başkaları tarafından da kullanılmasını sağlayarak, kurumların dışarı açılmalarını desteklerler.
- Kurumlara özel olarak yazılmış uygulamaları Web Servisler haline getirerek yazılımların ömrü uzatılmış olur.

Ancak anlamsal olarak zenginleştirilmediklerinde Web Servislerin kullanım alanının çok sınırlı olduğu yaygın olarak kabul edilmiş bir gerçektir. Bununla birlikte sağlık bilgi sistemleri HL7, CEN TC251, ISO TC215 (ISO TC215) ve GEHR gibi standartlar yoluyla çok büyük ölçüde ilgi alanı bilgisi sunan az alanlardan biridir. Bu projede bu bilgiler ontolojiler haline getirilip Web Servisleri anlamsal olarak zenginleştirmek amacıyla kullanılmıştır.

Bu projede önerilen Web Servis mimarisi global olarak üzerinde anlaşılan ontolojiler önermek yerine sağlık kurumları kullanılan standartlardaki anlamsal farlılıkları önerilen aracı (mediator) komponenti yardımıyla anlaşılabilir hale getirmeyi amaçlamıştır. Sağlık kurumları kendi ontolojilerini var olan standartları baz alarak tasarladıktan sonra, aracı (mediator) komponent da bu standart ontolojileri kullanarak ontoloji eşleştirme araçları yardımıyla anlamsal bütünlüğü sağlar. İkili anlamsal eşleşmeler en iyi Eşler-Arası (Peer-to-Peer) paradigmasını kullanarak modellenebildiğinden, aracılar arası iletişim P2P olarak tasarlanmıştır. P2P platformu olarak Sun tarafından geliştirilen JXTA (JXTA) altyapısı kullanılmıştır. Kullanılan P2P mimari, aynı zamanda Web Servis Depolarının da dağınık bir ortamda bulunmasını sağlamıştır. Böylece Medikal Web Servisleri P2P ortamda bir Web Servis deposunda da, tek peerlarda da bulunuyor olsalar anlamsal özellikleri aracılığıyla bulunabilmektedir. Proje çerçevesinde geliştirilmiş olan altyapıya Mobil araçlar tarafından da ulaşılmasını sağlayan kolay kullanılabilir arayüzler tasarlanmıştır. Böylece çoğunlukla mobil olarak çalışan sağlık personelinin Web Servis olarak sağlanan servislere mobil araçlarından da ulaşabilmeleri sağlanmıştır.

Anahtar kelimeler: Sağlık, Birlikte İşlerlik, Web Service, P2P, Mobil, Ontoloji, HL7, GEHR, CEN TC251 , ISO TC215

ABSTRACT

Most of the current Healthcare Information Systems are developed for special purposes and serve only certain departments of the Healthcare Organizations. Moreover, the healthcare record of a patient may be at different Healthcare Organizations that can not collaborate. All these make it difficult to Healthcare personnel to access all of the healthcare records of a patient. This project aims to develop a Web Service based interoperability platform on a P2P infrastructure as a solution to this problem.

The healthcare standards that are nowadays commonly used like CEN TC251, GEHR and HL7 are also trying to solve the interoperability problem of Healthcare Information Systems. However, although many systems are developed conforming to these standards, the interoperability problem in the healthcare sector is still unresolved due to the reasons listed below:

- First of all, some of the mentioned standards do not aim an interoperability at a level that will support a fully automatic data processing. For example, the Electronic Healthcare Record based standards ENV 13606 (CEN) and GEHR, have defined the meaningful parts of the healthcare records so that the patient records can be more understandable with the aid of a computer and better categorized.
- Even if systems are developed conforming to a standard, the interoperability problem with a sistem conforming to another standard cannot be resolved. Actually, the biggest problem of the unified healthcare information systems is the communication system between various standards.

Besides, The Web Service Model provides the healthcare industry an ideal platform to resolve the hard to solve interoperability problem. Web Services can provide the interoperability of different applications by wrapping existing software. The benefits of Web Services to the Healthcare domain can be summarized as follows:

- It enables the interoperability of Healthcare Information Systems by standardizing the access to records through WSDL and SOAP, instead of standardizing the documentation of Electronic Healthcare Records,
- The Healthcare Information Systems are confronted with the problem that the number of standards that are evolved to present the same data in different forms rapidly increase. Web Services enable the perfect interoperability of applications conforming to different or even competing standards.
- Web Services provide that the existing services of the Healthcare organizations can be used by others as well.
- Software lifetime is expanded by providing a Web Service interface to the applications that are peculiarly written for an organization.

However, it is a generally accepted fact that Web services have a limited domain when they are not semantically enriched. Nevertheless, Healthcare Information Systems are ampng the few domains that present considerable domain information through standards like HL7, CEN TC251, ISO TC215 and GEHR. In this project, these information are converted to ontologies and used to semantically enrich Web Services.

The Web Service architecture proposed in this project aimed to make the semantic differences of the standards used in healthcare organizations understandable with a mediator component instead of proposing gloabally accepted ontologies. After healthcare organizations design their own ontologies based on existing ontologies, the mediator component provides the semantic integrity using these standard ontologies with the help of the ontology mapping tools. As bipartite pairings are best modelled using the peer-to-peer paradigm, the communication between mediators is designed as P2P. The JXTA infrastructure developed by Sun Microsystems is used as the P2P platform. The P2P architecture used also provides that the Web Service repositories are distributed. This way Medical Web Services can be found with their semantic properties in the Peer-to-Peer environment no matter they are in a Web Service repository or on a single peer. Easy-to-use interfaces that enable access with Mobile tools to the infrastructure developed in the scope of the project are developed as well. Thus, it becomes possible that the healthcare personnel, mostly working mobile, can access services provided as Web Services from their Mobile tools.

Keywords: Healthcare, Interoperability, Web Service, P2P, Mobile, Ontology, HL7, GEHR, CEN TC251 , ISO TC215

PROJE ANA METNİ

GİRİŞ

Günümüzde her ne kadar birlikte işlerliği sağlamak amacıyla standartlar geliştirilse de, bunlar her zaman yeterli olmuyor. Sağlık bilgi sistemlerinde de durum benzer. Başlıca sorunlar şu şekilde sıralanabilir:

- birden fazla standardın var oluşu
- aynı standartlara uyan sağlık bilgi sistemlerinin dahi birlikte işler olamayışı
- sağlık kuruluşu içindeki birimler için bile birden fazla standarda ihtiyaç duyulması
- birimler arası birlikte işlerliğin eksikliği.

Bu bağlamda Web Servisler birlikte işlerliği sağlamak amacıyla önemli bir araç olmakla beraber, ancak tek başına yeterli değildir. Web servisler kullanıma açılsa bile, uygun bir Web Servise ihtiyaç duyulduğu zaman bulunması da ayrı bir problem teşkil etmektedir. Bu da Web Servislere anlamsal özellikler katarak ontolojiler vasıtasıyla gerçekleştirilebilir. Farklı standartlara uyan sağlık kurumlarını birlikte işler kılabilmek için ise, sağlık bilgi sistemlerini kendilerin standart ontolojileri baz alarak bir ontoloji oluşturması gerekmektedir. Bu projede tanımlanan ontolojilerin eşleştirmelerini sağlayıp, sağlık kurumlarının birlikte işlerliğini bir P2P altyapısı geliştirerek, ve bu altyapıya Web Servislerin konuşlandırılması için Web Servis depolarını entegre edip Web Servislerin kolayca bulunması için anlamsal mekanizmalar geliştirilmişitir. İlgili standartlarla olan ihtiyacı ve alakaları tanımlanıp, araştırmaya açık problemleri araştırılmıştır.

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 OWL'dir (OWL). OWL-S, OWL temelinde oluşturulmuştur.

HL7, CEN TC251, ISO TC215 ve GEHR sağlık bilgi sistemlerinin kullandıkları standartlardır. Bunlardan HL7 aradaki mesajlaşmaları standartlaşmayı hedeflemekte iken, GEHR elektronik hasta kayıtlarının standartlaşması üzerine çalışmaktadır.

ebXML (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.

Benzer bir şekilde UDDI (UDDI) saklayıcısı da Web Servisleri tutup uygun nitelikte olanların kolayca bulunmasına olanak sağlar.

JXTA Sun tarafından geliştirilen açık kaynak kodlu bir P2P altyapısı sunmaktadır. Bu kütüphane P2P ortamını yaratmak, aradaki mesajlaşmayı kolayca sağlamak ve P2P ortamında ihtiyaç olacak mekanizmaları ya da altyapılarını sağlamaktadır.

GEREÇ VE YÖNTEM

Proje süresince yürütülecek yönteme göre, ilk etapta geliştirilecek olan yazılım parçalarının analiz ve tasarımı yapıldı. Bu tasarım çerçevesinde sistemin temel özelliklerini taşıyan basit bir prototip yazılımı gerçekleştirildi, bu prototip irdelenerek tasarımdaki eksiklikler giderildi ve tasarım geliştirildi. 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 aktarıldı.

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

Başlıca Aşamalar	**Ayrıntılı Bilgi**	**Zamanlama (ay)**
Medikal Ontolojilerin Geliştirilmesi	Var olan HL7, CEN ENV 13606, GEHR standartlarından faydalanarak proje dahilinde kullanılacak Medikal Ontolojiler geliştirilmiştir.	1-2
Medikal Web Servislerin kolayca geliştirililmesi	Medikal Web Servislerin kolayca geliştirilmesini ve anlamsal olarak zenginleştirilmesini sağlayacak uygulamalar geliştirilmiştir.	3-4
Medikal Ontolojilerin Eşleştirilmesi için bir Platform geliştirmek	Farklı standartlara uyumlu geliştirilmiş sağlık bilişim sistemlerinin "birlikte işlerliğini" sağlamak amacıyla farklı ontolojilerin birbirlerine çevrilmesini sağlayacak Ontoloji Eşleme platformu geliştirilmiştir. Böylece bir sağlık sistemi kendi ontolojisi ile tanımlanmamış bir Web Servisi de P2P mimaride arayıp bulup sorunsuz bir şekilde kullanabilmektedir.	5-6
P2P mimarisinde varolan arama mekanizmalarını geliştirmek	Anlamsal olarak zenginleştirilmiş Web Servislerin sorgulanmasını sağlamak için P2P sistemlerin varolan "anahtar kelime" bazlı arama sistemleri geliştirilmiştir.	7-8
Web Servis Depolarının P2P mimariye entagrasyonu	Web Servis depolarının P2P arama mekanizmaları yoluyla sorgulanması sağlanmış, böylece Web Servis depolarının da bulunması otamatize edilmiştir.	9-10
Tüm sistemin entegrasyonu ve Mobil arayüzlerin geliştirilmesi	Geliştirilen tüm komponentlerin entegrasyonu gerçekleştirilmiş ve sisteme mobil araçlar için hazırlanmış arayüzler aracılığıyla erişim saplanmıştır.	11-12

BULGULAR

1) *Medikal Ontolojilerin Geliştirilmesi:* Bu iş tabloda gösterildiği gibi projenin ilk iki ayını kapsamaktadır. Bu iş başarıyla tamamlanmıştır. Belirtilen bu iş çerçevesinde, proje dahilinde kullanılmak üzere gerekli alan araştırılması yapılıp, ontoloji geliştirme araçlarının da yardımlarıyla, projeye özgü medikal ontolojiler geliştirilmiştir. Bu ontolojiler temel olarak medikal kurumların uzmanlığını belirtmede kullanılacak olup, çeşitli birçok uzmanlığı kapsayacak şekilde tasarlanmıştır. Ayrıca bu ontolojiler, medikal servislerin girdi ve çıktılarını belirtmede de kullanılmıştır. Geliştirilen medikal uzmanlik, Web servis fonksiyonalite, ve klinik konsept ontolojilerinin birer örneğine asagida siralalan adreslerden ulaşılabilinir.
   - http://www.srdc.metu.edu.tr/~yildiray/tubitak/ExpertiseOnt.rdfs
   - http://www.srdc.metu.edu.tr/~yildiray/tubitak/HL7FuncOnt.owl
   - http://www.srdc.metu.edu.tr/~yildiray/tubitak/CCOCPT.rdfs
   - http://www.srdc.metu.edu.tr/~yildiray/tubitak/CCOBUT.rdfs
   - http://www.srdc.metu.edu.tr/~yildiray/tubitak/CCOLOINC.rdfs

2) *Medikal Web Servislerin kolayca Geliştirilmesi:* Bu iş tabloda görüldüğü gibi projenin 3. ve 4. ayını kapsamaktadır. Bu bölüm kapsamında, Web Servisleri online bir şekilde anlamsal olarak zenginleştirip, tüm P2P ortamına duyurulmasını sağlayacak araçlar geliştirilmiştir. Geliştirilen bu araçlar Java'nın Servlet teknolojisine sahip olup, tüm Web ortamından erişilebilinmektedir. Kullanıcıya, Servlet teknolojisi ile bir arayüz sağlayan bu araç arka planda JXTA teknolojisini kullanarak, P2P ortamla gerekli iletişimi sağlamaktadır.

```
Admit Patient
Input
        Name (string)
        Surname (string)
        Sex (string)
        Ambulatory Status
(string)
Output
        Status: ACCEPTED or
Reason To reject (string)
        Local Patient ID
(integer)
```

```
Retrieve Surgery Results
Input
        Local Patient ID (integer)
Output
        Status: OK or Reason to reject
        (string)
        Result Count (integer)
        (multiple)
        Operation Code (Source CPT,
Destination BUT) (string)
        Operation Description (string)
        Date of Operation (date)
        Document Accession Number
        (integer)
```

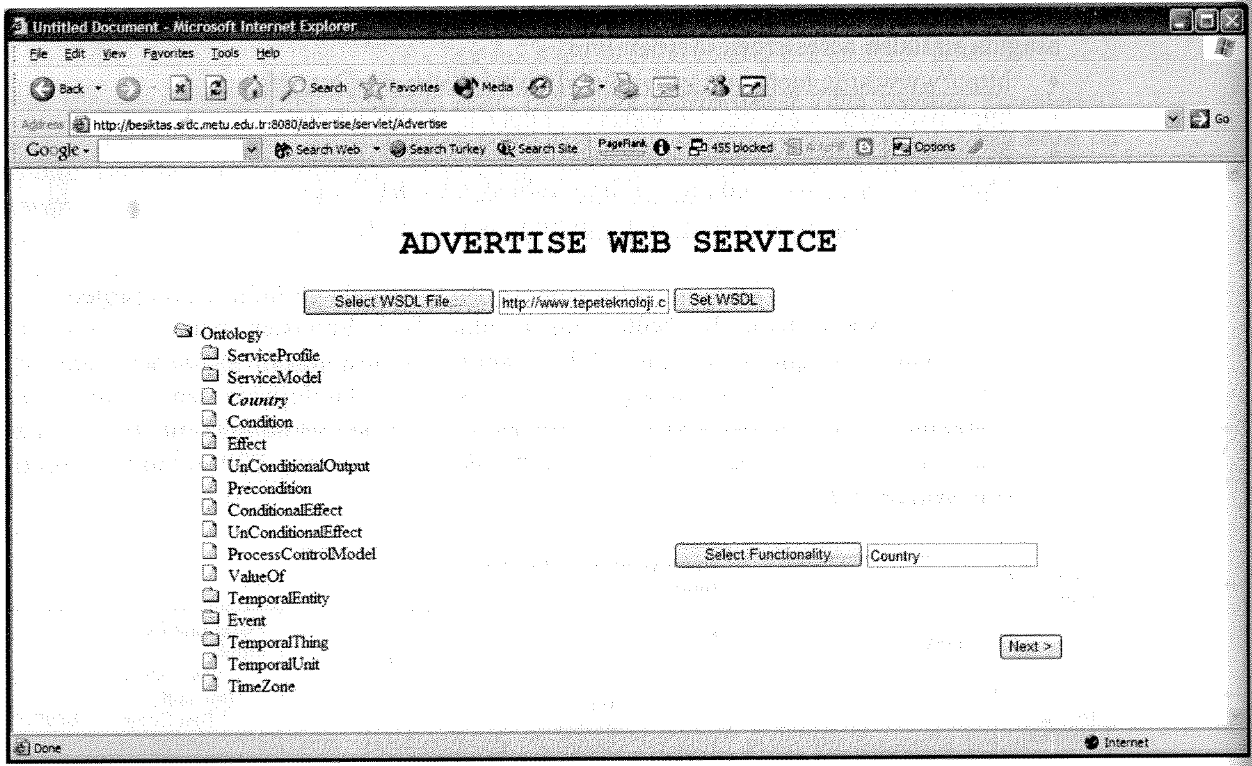
```
Place Lab Order
Input
    Local Patient ID (integer)
    (multiple)
      Test Code (LOINC/CPT)
      (string)
Output
     Process ID (integer)
```

```
Retrieve Document
Input
      Document Accession
Number (integer)
Output
      Document MIMETYPE
      (string)
      Binary Output
      (base64binary)
```

```
Push Lab Results
Input
        Process ID (integer)
        Result Count (integer)
(multiple)
        Test Code (LOINC/CPT) string
        Result (string)
        Normal Values (string)
        Date of Result (date)
        Time of Result (time)
Output
        Status: OK or Error (string)
```

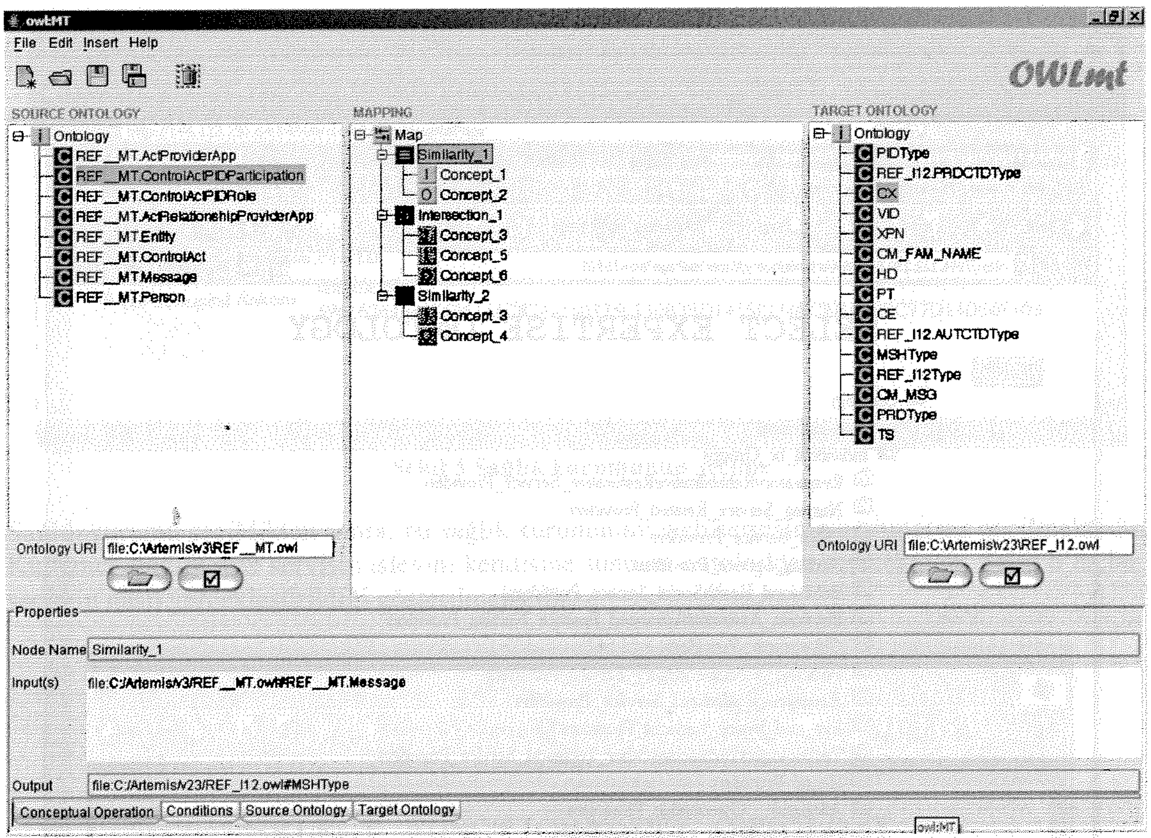
Şekil 1 Geliştirilen Web Servislerin Yapısı

Şekil 1'de geliştirilen Web Servislerin yapıları görülebilmektedir. Şekil 2 ise Web Servislerin ortama tanıtıldığı web arayüzünü göstermektedir. İlgili Web Servisler bu arabirim ile WSDL konumları ve ilgili işlevi seçilerek ortama tanıtılır.



Şekil 2 Web Servis Bildirim Ekranı

3) *Medikal Ontolojilerin Eşleştirilmesi için bir Platform geliştirmek:* Bu iş tabloda görüldüğü gibi projenin 5. ve 6. ayını kapsamaktadır. Bu iş dahilinde, iki adet medikal ontolojiyi eşleştirecek platform, tüm gerekli özellikleri ile tamamlanmıştır (OWLmt). Şekil 3'den de görülebileceği gibi, seçilen iki ontoloji üzerinde, ilgili ontoloji dallarında kesişim, birleşim, benzerlik ve buna benzer birçok operasyon yardımıyla iki ontoloji eşleştirilebilinir. Eşleştirilme işlemi başarı ile bitirildikten sonra, bir ontoloji için verilen bir örnekten diğer ontolojiye uygun örnek elde edilebilinir.



Ontology URI file:C:\Artemis\v3\REF__MT.owl

Ontology URI file:C:\Artemis\v23\REF_I12.owl

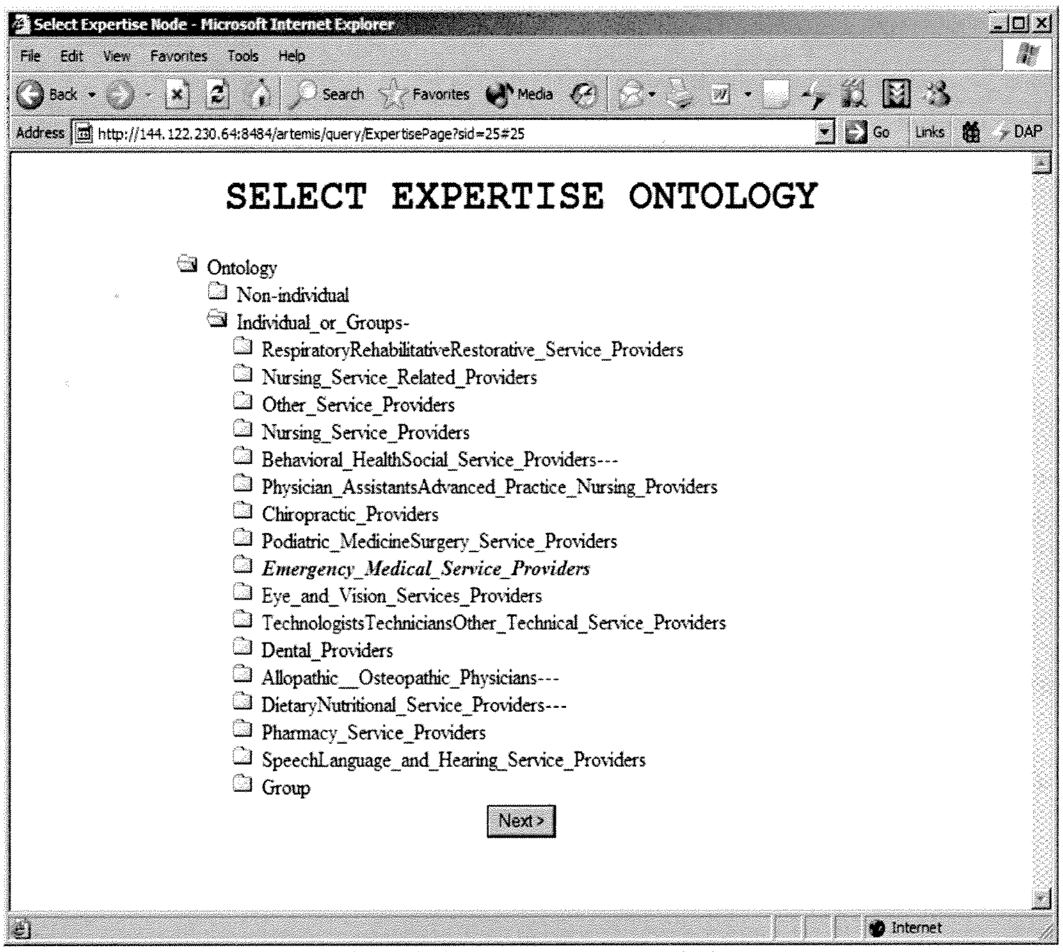
Node Name Similarity_1

Input(s) file:C:/Artemis/v3/REF__MT.owl#REF__MT.Message

Output file:C:/Artemis/v23/REF_I12.owl#MSHType

Şekil 3 OWLmt

4) *P2P mimarisinde varolan arama mekanizmalarını geliştirmek:* Anlamsal olarak zenginleştirilmiş Web Servislerin sorgulanmasını sağlamak için P2P sistemlerin varolan "anahtar kelime" bazlı arama sistemleri geliştirilmistir. Bu gelişim sayesinde P2P ortamında daha etkili ve verimli aramalar yapılabilmektedir.


Select Expertise Node - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://144.122.230.64:8484/artemis/query/ExpertisePage?sid=25#25
Go Links DAP
Internet

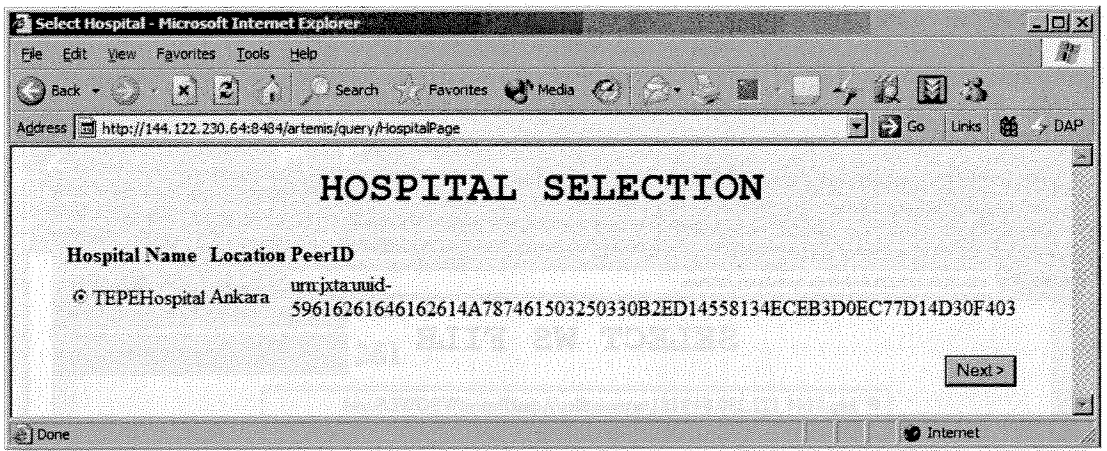
SELECT EXPERTISE ONTOLOGY

Ontology
Non-individual
Individual_or_Groups-
RespiratoryRehabilitativeRestorative_Service_Providers
Nursing_Service_Related_Providers
Other_Service_Providers
Nursing_Service_Providers
Behavioral_HealthSocial_Service_Providers---
Physician_AssistantsAdvanced_Practice_Nursing_Providers
Chiropractic_Providers
Podiatric_MedicineSurgery_Service_Providers
Emergency_Medical_Service_Providers
Eye_and_Vision_Services_Providers
TechnologistsTechniciansOther_Technical_Service_Providers
Dental_Providers
Allopathic__Osteopathic_Physicians---
DietaryNutritional_Service_Providers---
Pharmacy_Service_Providers
SpeechLanguage_and_Hearing_Service_Providers
Group

Next >

Şekil 4 Uzmanlık alanının seçimi

Şekil 4'te görüldüğü üzere, öncelikle bir sağlık kurumu aranmaktadır. Bu sağlık kurumu, uzmanlık alanına göre aranmaktadır. Bu uzmanlık alanına uygun bulunan sağlık kurumları kullanıcıya Şekil 5'teki gibi sunulmaktadır. Kullanıcı sunulan seçeneklerden dilediğini seçip bir sonraki adima geçmektedir.

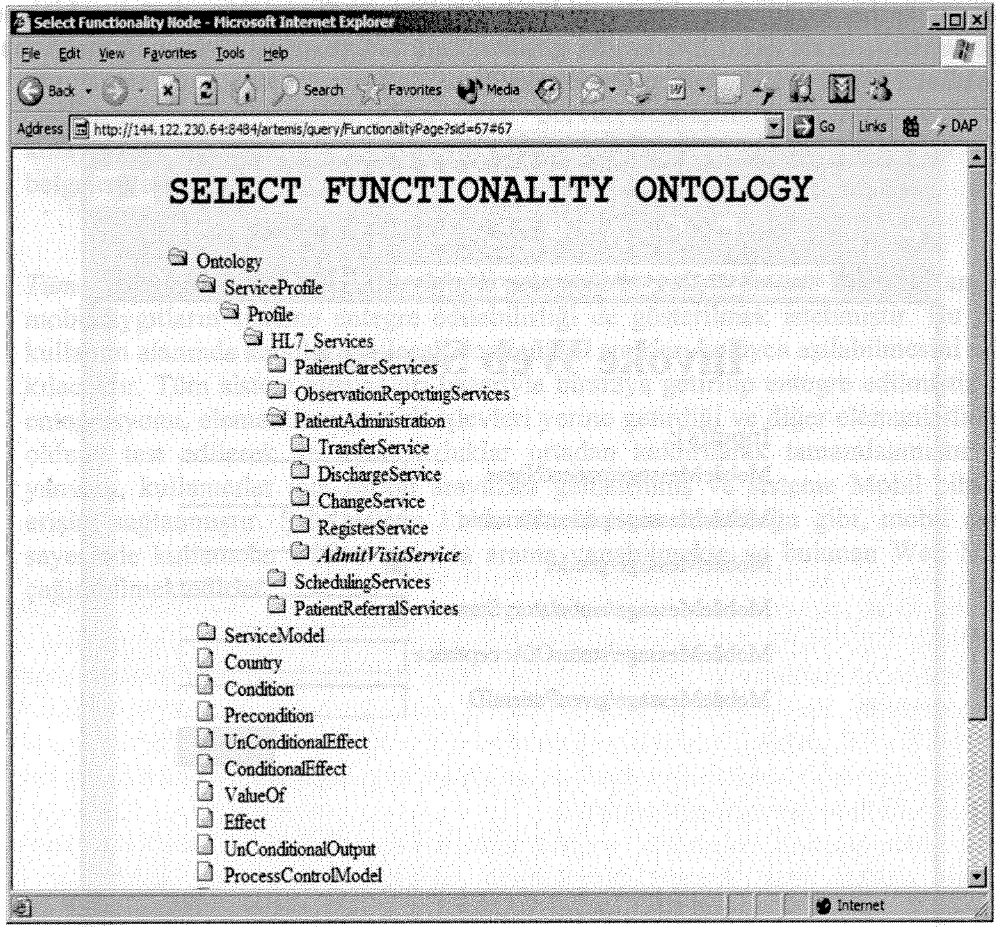


HOSPITAL SELECTION

Hospital Name	Location	PeerID
◉ TEPEHospital	Ankara	urn:jxta:uuid-59616261646162614A787461503250330B2ED14558134ECEB3D0EC77D14D30F403

Şekil 5 Sağlık kurumunun seçimi

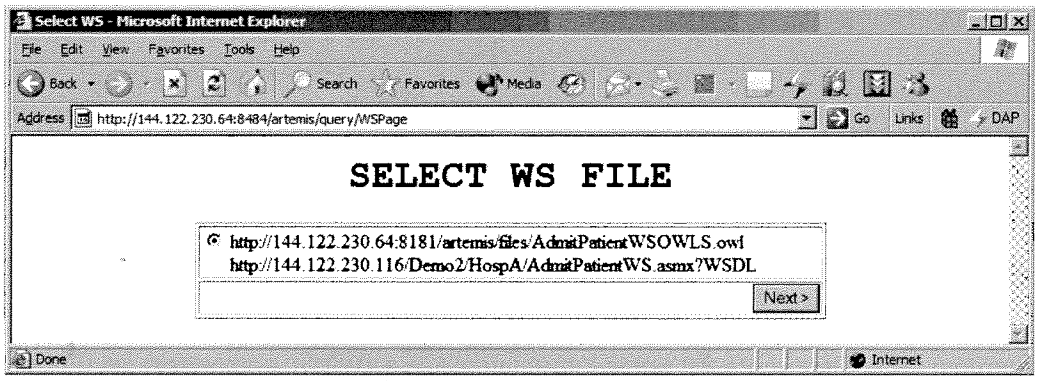
Sağlık kurumu seçildikten sonra, bu sağlık kurumunun web servisinin aranmasına geçilmektedir. Kullanıcı aradığı web servisin işlevini kendisine sunulan ontolojiden seçmektedir. Şekil 6'da bu seçimin nasıl gerçekleştirildiği gösterilmektedir.



SELECT FUNCTIONALITY ONTOLOGY

- Ontology
  - ServiceProfile
    - Profile
      - HL7_Services
        - PatientCareServices
        - ObservationReportingServices
        - PatientAdministration
          - TransferService
          - DischargeService
          - ChangeService
          - RegisterService
          - *AdmitVisitService*
        - SchedulingServices
        - PatientReferralServices
  - ServiceModel
  - Country
  - Condition
  - Precondition
  - UnConditionalEffect
  - ConditionalEffect
  - ValueOf
  - Effect
  - UnConditionalOutput
  - ProcessControlModel

Şekil 6 Aranan servis işlevinin seçilmesi

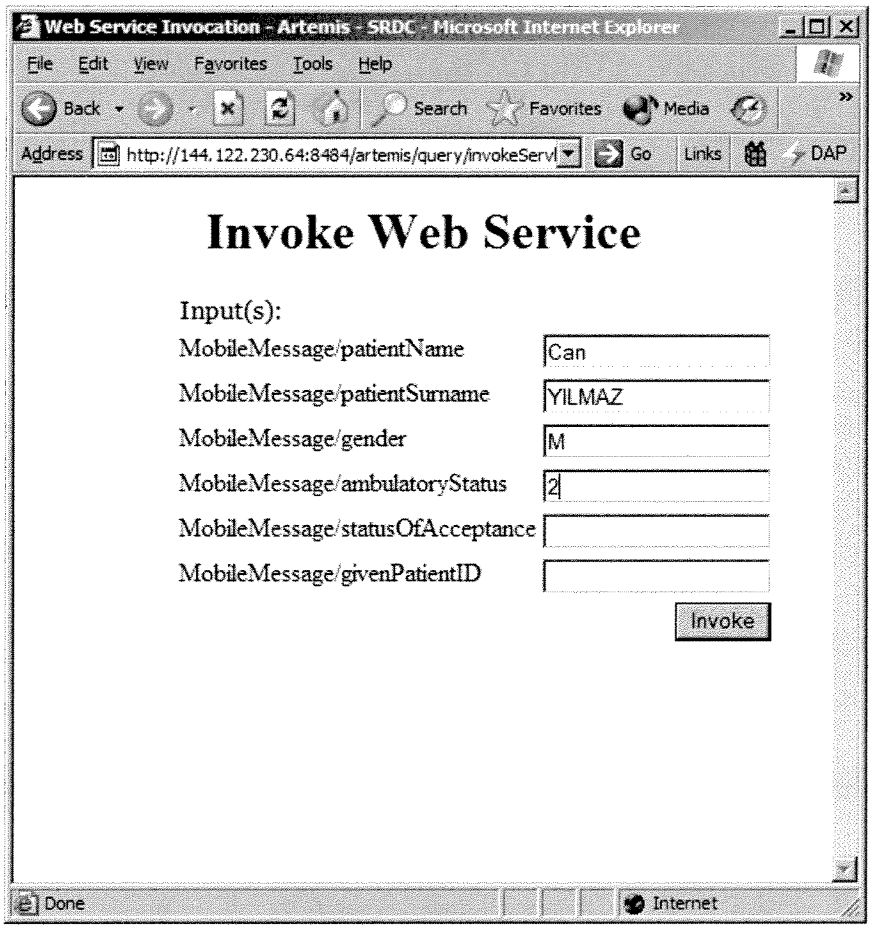

Select WS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://144.122.230.64:8484/artemis/query/WSPage
Go Links DAP

SELECT WS FILE

- (●) http://144.122.230.64:8181/artemis/files/AdmitPatientWSOWLS.owl
- ( ) http://144.122.230.116/Demo2/HospA/AdmitPatientWS.asmx?WSDL

Şekil 7 Servisin seçilmesi

Şekil 7'de görüldüğü gibi, seçilen işleve uygun web servisler kullanıcıya sunulmaktadır. Kullanıcı bunlardan birini seçerek, seçtiği Web servisin girdilerini girmek üzere bir sonraki adıma geçer. Bu adımda Şekil 8'deki gibi girdileri belirttikten sonra, servis çağrılır. Sonuç bir sonraki ekranda (Şekil 9) kullaıcıya gösterilmektedir.

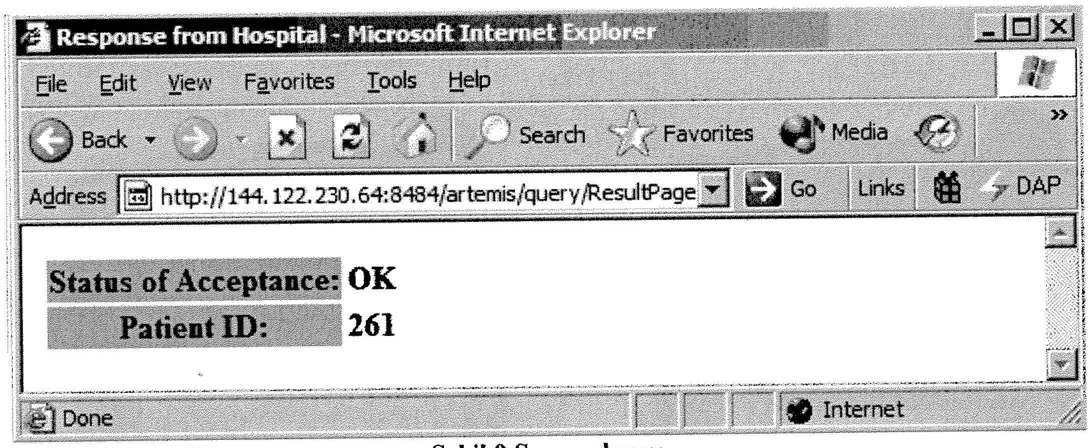

Web Service Invocation - Artemis - SRDC - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://144.122.230.64:8484/artemis/query/invokeServi
Go Links DAP

Invoke Web Service

Input(s):

MobileMessage/patientName	Can
MobileMessage/patientSurname	YILMAZ
MobileMessage/gender	M
MobileMessage/ambulatoryStatus	2
MobileMessage/statusOfAcceptance	
MobileMessage/givenPatientID	

Şekil 8 Servisin çağrılması

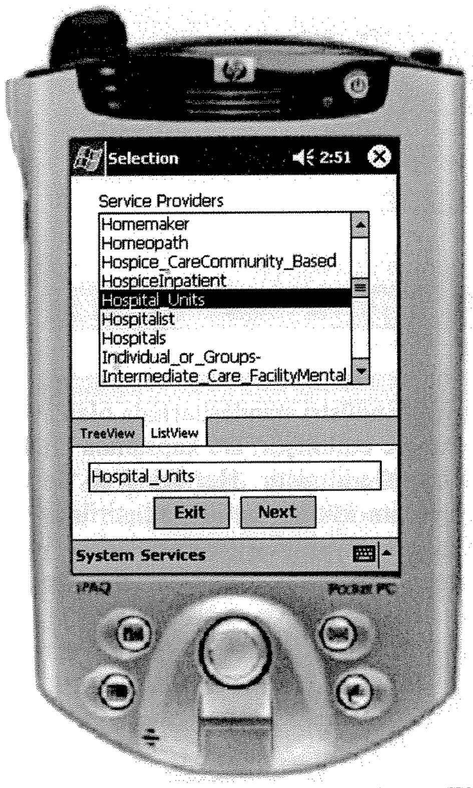

Response from Hospital - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://144.122.230.64:8484/artemis/query/ResultPage Go Links DAP
Done Internet

Status of Acceptance: OK
Patient ID: 261

Şekil 9 Sonuç ekranı

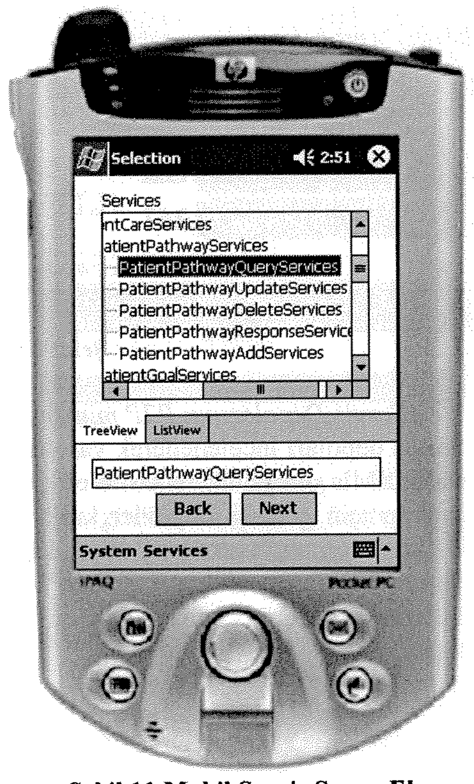
5) *Web Servis Depolarının P2P mimariye entagrasyonu:* Endüstri standartlarında olan Web Servis depoları incelenenerek varolan mimariye entegre edilmiştir. Bu kapsamda UDDI ve ebXML olmak üzere iki Web Servis deposu tespit edilmiştir. Her iki Web Servis deposu için gerekli arayüzler kullanılarak, kolay erişim kütüphaneleri geliştirilmiş ve bunlardan yararlanılmıştır. Bu kütüphaneler sayesinde Web Servis depolarina Web Servis tanıtmak, arama yapmak kolaylaştırılmıştır. Tasarımın esnek tutulup ileride endüstrinin kabul edeceği diğer Web Servis depolarının da entagrasyonuna da kolaylıklar sağlanıp projenin uygulama ömrünün uzatılması için gerekli belgelendirmeler yapılmıştır.

6) *Tüm sistemin entegrasyonu ve Mobil arayüzlerin geliştirilmesi:* Tanımlanan projede mobil aygıtların sisteme entegre edilebilirliği de gösterilmek istenmiştir. Bu projenin kullanım alanında karşılaşılabilecek donanımsal sınırları kolayca aşılabilmesini mümkün kılacaktır. Tüm sistem elemanları başarıyla biraraya getirilip entegre edilmiştir. Sistem entegrasyonu, elemanların gerekli işlevleri yerine getirdiği ve diğer elemanlarla uyumlu olduğu test edilerek ve uyumsuzluklar ortadan kaldırılarak tamamlanmıştır. Bunun yanısıra, kullanıcılar için Mobil arayüzler geliştirilmiş ve sisteme Mobil cihazlar ile erişim sağlanmıştır. Şekil 10 ve 11'de örneklerinin sunulduğu gibi, mobil arayüzler sayesinde kullanıcılar P2P ortamında arama yapabilmekte ve bulunan Web Servisleri çağırabilmektedirler.



Hospital_Units

Şekil 10 Mobil Organizasyon Sorgu Ekranı


Selection 2:51
Services
ntCareServices
atientPathwayServices
PatientPathwayQueryServices
PatientPathwayUpdateServices
PatientPathwayDeleteServices
PatientPathwayResponseServic
PatientPathwayAddServices
atientGoalServices
TreeView
ListView
Back
Next
System Services
iPAQ
Pocket PC

PatientPathwayQueryServices

Şekil 11 Mobil Servis Sorgu Ekranı

TARTIŞMA/SONUÇ

Proje başarıyla tamamlanmıştır. Geliştirilen sistem ile, sağlık kurumları varolan ontojileri baz alarak kendi ontolojilerini tanimladiktan sonra, hizmetlerini Web Servis olarak dışarı açabilecek ve diğer sağlık kurumlarının Web Servislerine erişebilecekler. Tanımlanan ontojiler ve aracı bileşenin yardımıyla sağlık kurumları arası iletişim farklı standardlara uysalar bile gerçekleşmektedir. Bu kapsamda, Web Servis depolari P2P ortamina entegre edildi, Medikal ontojiler geliştirildi, ontolojilerin eşleştirilebilmesi için araçlar üretildi, ve Medikal uygulamaları kolayca Web Servis ile erişilebilir kılmak amacıyla bir uygulama geliştirilmiştir. Böylece sağlık kurumlarının birlikte işlerliği sağlanmıştır.

Bu projede yapılan çalışmalar sonucu elde edilen bilgiler ve yazılımlar kurumumuz bünyesinde devam eden Avrupa projelerinde kullanılmıştır. Ayrıca, çalışmalar bilimsel araştırmalar ışığında zenginleştirilmiş ve sonucunda aşağıdaki Yüksek Lisans tezleri hazırlanmıştır:

- "Design and Implementation of Semantically Enriched Web Services in the Healthcare Domain", Umit Lutfu Altintakan, M.S. Thesis, Dept. of Computer Eng., December 2004.

- "Developing JXTA Applications for Mobile Devices and Invoking Web Services Deployed in JXTA Platform from Mobile Devices", Mesut Bahadir, M.S. Thesis, Dept. of Computer Eng., December 2004.

Proje öneri formunda belirtilen amaç ve kapsama uygun olarak sonuçlanmıştır.

PROJE ÖZET BİLGİ FORMU

Proje Kodu : 104E013

Proje Başlığı :
SAĞLIK BİLGİ SİSTEMLERİ İÇİN WEB SERVİSLERİ TABANLI BİR P2P BİRLİKTE İŞLERLİK PLATFORMU
(A WEB SERVICE BASED P2P INTEROPERABILITY PLATFORM FOR MEDICAL INFORMATION SYSTEMS)

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

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

Destekleyen Kuruluş(ların) Adı ve Adresi :

Projenin Başlangıç ve Bitiş Tarihleri : 01/06/2004 – 01/06/2005

Öz : (en çok 70 kelime)

Proje sağlık bilgi sistemlerinin birlikte işlerliğini sağlamayı amaçlamaktadır. Bu kapsamda sağlık kurumları hizmetlerini anlamsal olarak zenginleştirilmiş Web servisler olarak açmaktadırlar. Geliştirilen P2P altyapısına Web Servis depolarının entegre edilmesi ile ihtiyaç olan Web servislere anlamsal özelliklerini de kullanılarak kolayca erişilmektedir. Sağlık kurumların ve hatta birimlerinin farklı standartlara uyması kaçınılmaz olduğu için aradaki mesajlaşmalar bir aracı bileşen sayesinde ontolojilerin eşleştirilmesi ile tarafların anlayabilecekleri şekle çevrilmektedir.

Anahtar Kelimeler:
Sağlık, Birlikte İşlerlik, Web Service, P2P, Mobil, Ontoloji, HL7, GEHR, CEN TC251 , ISO TC215

Projeden Kaynaklanan Yayınlar :

1. Dogac, A., Laleci, G., Kirbas S., Kabak Y., Sinir S., Yildiz A., Gurcan Y. "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain", Information Systems Journal (Elsevier) special issue on Semantic Web and Web Services, accepted for publication. September 2004 (Science Citation Index Core, Impact Factor: 03.327)
2. A. Dogac, Y. Kabak, G. B. Laleci, C. Mattocks, F. Najmi, J. Pollock, "Enhancing ebXML Registries to Make them OWL Aware", accepted for publication in the Distributed and Parallel Databases Journal, Kluwer Academic Publishers.

3. Dogac, A., Laleci, G., Kabak, Y., Unal, S., Beale, T., Heard, S., Elkin, P., Najmi, F., Mattocks, C., Webber, D., "Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics" submitted for publication upon invitation, International Journal of Metadata, Semantics and Ontologies
4. Dogac, A., Bussler, C., A Tutorial on Providing Interoperability of Medical Information Systems through Semantically Enriched Web Services ", The Fourth International Conference on Web Engineering (ICWE 04), Munich, Germany, July 28-30, 2004.
5. Bicer, V.,Laleci, G.,Dogac, A., Kabak, Y., "Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain" ACM Sigmod Record, Vol. 34, No. 3, September 2005. (Science Citation Index Expanded, Impact Factor: 00.675)
6. Bicer, V.,Laleci, G., Dogac, A., Kabak, Y.,"Providing Semantic Interoperability in the Healthcare Domain through Ontology Mapping", accepted for publication in eChallenges 2005, Ljubljana, Slovenia.

Bilim Dalı :

Doçentlik B. Dalı Kodu :

REFERANSLAR

[HL7] HL7, http://www.hl7.org/
[GEHR] GEHR, www.openehr.org/
[CEN TC251] CEN TC251, http://www.centc251.org/
[ISO TC215] ISO TC215,
http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=529137&objAction=browse&sort=name
[ebXML] ebXML, http://www.ebxml.org/
[UDDI] UDDI, http://www.uddi.org/
[OWL] OWL, http://www.w3.org/TR/owl-features/
[JXTA] JXTA Project, http://www.jxta.org/
[WSDL] WSDL, http://www.w3.org/TR/wsdl
[SOAP] SOAP, http://www.w3.org/TR/wsdl
[OWLmt] OWLmt, http://www.srdc.metu.edu.tr/webpage/projects/artemis/owlmt

Distributed and Parallel Databases, 18, 9–36, 2005

Enhancing ebXML Registries to Make them OWL Aware*

ASUMAN DOGAC asuman@srdc.metu.edu.tr
YILDIRAY KABAK yildiray@srdc.metu.edu.tr
GOKCE B. LALECI banu@srdc.metu.edu.tr
Software Research and Development Center, Department of Computer Engineering, Middle East Technical University (METU), 06531, Ankara, Turkiye

CARL MATTOCKS carlmattocks@checkmi.com
CHECKMi, 225 Emerson Lane, Berkeley Heights, NJ 07922, USA

FARRUKH NAJMI farrukh.najmi@sun.com
Sun Microsystems, Inc., 1 Network Dr., Burlington, MA 01810, USA

JEFF POLLOCK jeff.pollock@networkinference.com
Cerebra, Inc., 5963 La Place ct., Carlsload, CA 92008, USA

Recommended by: Athman Bouguettaya and Boualem Benatallah

Abstract. ebXML is a standard from OASIS and UN/CEFACT which specifies an infrastructure to facilitate electronic business. In this paper, we address how ebXML registry semantics support can be further enhanced to make it OWL aware. OWL constructs are represented through ebXML registry information model constructs, and stored procedures are defined in the ebXML registry for processing the OWL semantics. These predefined stored queries provide the necessary means to exploit the enhanced semantics stored in the registry. In this way, an application program does not have to be aware of the details of how this semantics support is achieved in ebXML registry, and does not have to contain additional code to process this semantics.

We believe that this approach is quite powerful to associate semantics with registry objects: it becomes possible to retrieve knowledge through queries, the enhancements to the registry are generic and also the registry specification is kept intact. The capabilities provided move the semantics support beyond what is currently available in ebXML registries and it does so by using a standard ontology language.

To be able to demonstrate the benefits of the enhancements, we also show how the resulting semantics can be made use of in Web service discovery and composition.

Keywords: semantics, ebXML registry, Web Ontology Language (OWL), semantic web service discovery

*This work is supported in part by the European Commission, Project No: IST-1-002104-STP SATINE and the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 104E013 and is realized as a proposal to OASIS ebXML Semantic Content Management subcommittee which is working on possible semantic extensions to the registry.

1. Introduction

Electronic Business XML (ebXML) [14] is a standard from OASIS [27] and United Nations Centre for Trade Facilitation and Electronic Business, UN/CEFACT [45]. ebXML provides an infrastructure that allows enterprises to find each other's services, products, business processes, and documents in a standard way and thus helps to facilitate conducting electronic business. One of the important characteristics of ebXML compliant registries is that they provide mechanisms to define and associate semantics with registry objects. The ebXML "Registry" component holds the metadata for the registry objects and the documents pointed at by the registry objects reside in an ebXML repository. The basic semantic mechanisms of ebXML Registry are classification hierarchies consisting of classification nodes and the predefined associations among classification nodes. These are all registry objects and registry objects can be assigned properties through a slot mechanism. Given these constructs, considerable amount of semantics can be defined in the registry.

However, currently semantics is becoming a much broader issue than it used to be since several application domains are making use of ontologies to add the knowledge dimension to their data and applications [9, 17, 42]. One of the driving forces for ontologies is the Semantic Web initiative [2]. As a part of this initiative, W3C's Web Ontology Working Group defined Web Ontology Language (OWL) [31]. Naturally, there is lot to be gained from using a standard ontology definition language, like OWL, to express semantics in ebXML registries.

In this paper, we investigate how ebXML registries can be made OWL aware. There are three alternatives to support OWL ontologies through ebXML registries (Table 1):

Table 1. Three approaches to support OWL Ontologies through ebXML registries.

Method	Advantages	Disadvantages
–Representing OWL semantic constructs through ebXML Registry constructs	–No changes in the registry architecture specification and implementation –OWL semantics stored in an ebXML registry can be retrieved from the registry through native ebXML query facilities written by the user	–Further processiong needs to be done by the application program to make use of the enhanced semantics
–Providing predefined procedures to process OWL semantics in ebXML registry	–The user can call stored procedures when the need arises –The stored procedures can also be called transparently to the user by changing only the query manager component of the registry	–Limited reasoning capabilities
–Changing ebXML registry specification to support OWL with full reasoning capabilities	–Supports OWL with full reasoning capabilities –Makes it possible to deduce new data that is not directly stored in the registry	–Requires considerable changes in the registry architecture –Brings about the efficiency considerations of rule based systems

- Various constructs of OWL can be represented by ebXML classification hierarchies with no changes in the registry architecture specification and implementation. In this way, although some of the OWL semantics stored in an ebXML registry can be retrieved from the registry through ebXML query facilities, further processing needs to be done by the application program to make use of the enhanced semantics. For example, we can introduce "subClassOf" "association" to the ebXML registry to handle OWL multiple inheritance. Yet since ebXML registry does not natively support such an association type, to make any use of this semantics, the application program must have the necessary code, say, to find out all the super classes of a given class.
- To improve on the first alternative, the code to process the OWL semantics can be defined through generic stored procedures and be made available from the ebXML registry. For example, to find the super classes of a given class (defined through a new association type of "subClassOf"), a generic stored procedure can be defined. The user can call this procedure when the need arises. Furthermore, the stored procedures can also be called transparently to the user by the query manager. This involves an update in the query manager component of the registry. We believe that this approach is quite powerful to associate semantics with registry objects: it becomes possible to retrieve knowledge through queries and the enhancements to the registry are generic. Hence we take this approach.
- The third approach is changing the ebXML registry architecture to support OWL with full reasoning capabilities. Reasoning entails the derivation of new instances that are not directly stored in the registry. To deduce this data, rules need to be stored in the registry. However, this approach requires considerable changes in the registry architecture and brings about the efficiency considerations of rule based systems. Since our aim is to make ebXML registry OWL aware rather than specifying a new registry architecture, this approach will not be pursued any further in this paper.

There is another decision to be made: OWL is defined as three different sublanguages: *OWL Full*, *OWL DL* and *OWL Lite*, each geared towards fulfilling different requirements [25]. We choose *OWL Lite* since ebXML registries are for industrial use. Currently, the industry is using taxonomies like Universal Standard Products and Services Classification (UNSPSC) [46] and North American Industrial Classification Scheme (NAICS) codes for semantic descriptions and *OWL Lite* provides a quick migration path for taxonomies [41]. For *OWL Full*, it is unlikely that any reasoning software will be available [25]. *OWL DL* constructs, on the other hand, can be exploited best through the reasoners and as we have previously mentioned, using a reasoner natively in the registry requires changes in the ebXML registry architecture specification and this is not our purpose. Yet, for those who wish to represent *OWL Full* or *OWL DL* constructs, the mechanisms we present for OWL Lite can easily be adopted.

For ebXML registries, being OWL aware entails the following:

- Representing OWL constructs through ebXML Registry Information Model (RIM) constructs: For this purpose we show how OWL constructs can be expressed through ebXML registry semantic constructs.
- Automatically generating ebXML constructs from the OWL descriptions and storing the resulting constructs into the ebXML registry: We developed a tool to create an ebXML

Classification Hierarchy from a given OWL ontology automatically by using the transformations described in Section 4.

- Facilitating the querying of the registry for enhanced semantics: ebXML allows to annotate registry objects with classification nodes and ebXML query facilities allow these explicit relationships to be queried. However, when we introduce OWL semantics to the registry, there is a need for application code to do the necessary processing. For example, we may define a number of classes to be equivalent and we may wish to retrieve the instances of all the equivalent classes in the registry. In ebXML, to do this, a user must issue a number of seperate queries, using the result of a previous query as an input to the next query. To overcome this burden on the user side, we have provided a number of generic stored procedures to be invoked by the user when necessary. Note that these procedures can be invoked transparently to the user by making the necessary changes in the query manager component of the ebXML registry.

Furthermore, we show how the resulting semantics can be made use of in Web service discovery and composition.

This work is realized within the scope of IST-2104 SATINE project [23] as a proposal to OASIS ebXML Semantic Content Management subcommittee which is working on possible semantic extensions to the registry.

The paper is organized as follows: Section 2 briefly summarizes the main technologies involved in this work, namely, OWL and ebXML Registry architecture. In Section 3, we give an overall view of the approach and describe how the proposed enhancements fit into ebXML architecture. Section 4 describes how semantics defined in OWL ontologies can be represented and accessed in ebXML registries. Section 5 gives the related work. In this secion, we also provide a summary of the ebXML semantic standardization efforts undertaken by the OASIS open source standards body. Finally, Section 6 concludes the paper and presents the future work.

2. OWL and ebXML RIM

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)*

Web Ontology Language (OWL) is a semantic markup language for publishing and sharing ontologies on the World Wide Web [31]. OWL is derived from the DAML+OIL Web Ontology Language [1, 7] and builds upon the Resource Description Framework (RDF) [37, 38].

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*, and *owl:hasValue*.

RDF Schema Features	**(In)Equality**	**Property Characteristics**
– *Class (Thing, Nothing)*	– *equivalentClass*	– *ObjectProperty*
– *rdfs: subClassOf*	– *equivalentProperty*	– *DatatypeProperty*
– *rdf:Property*	– *sameAs*	– *inverseOf*
– *rdfs: subPropertyOf*	– *differentFrom*	– *TransitiveProperty*
– *rdfs:domain*	– *AllDifferent*	– *SymmetricProperty*
– *rdfs:range*	– *distinctMembers*	– *FunctionalProperty*
– *individual*		– *InverseFunctionalProperty*
Property Restrictions	**Restricted Cardinality**	**Datatypes**
– *Restriction*	– *minCardinality*	–*xsd datatypes*
– *onProperty*	– *maxCardinality*	**Class Intersection**
– *allValuesFrom*	– *cardinality*	–*intersectionOf*
– *someValuesFrom*		

Figure 1. OWL lite constructs.

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 [19]. 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,* and *owl:-inverseFunctionalProperty*. OWL constructs are given in more detail in Section 4.1 while describing their representation in ebXML registry.

OWL provides three decreasingly expressive sublanguages [41]:

- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. It is unlikely that any reasoning software will be able to support complete reasoning for *OWL Full* [25].
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). *OWL DL* is so named due to its correspondence with description logics which form the formal foundation of OWL.
- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints.

Within the scope of this paper, we consider *OWL Lite* constructs which are given in figure 1 and in the rest of the paper, OWL is used to mean *OWL Lite* unless otherwise stated.

2.2. *ebXML registry architecture and information model*

An ebXML registry consists of both a registry and a repository. The repository is capable of storing any type of electronic content, while the registry is capable of storing metadata that describes content. The content within the repository is referred to as "repository items" while the metadata within the registry is referred to as "registry objects". Clients access the

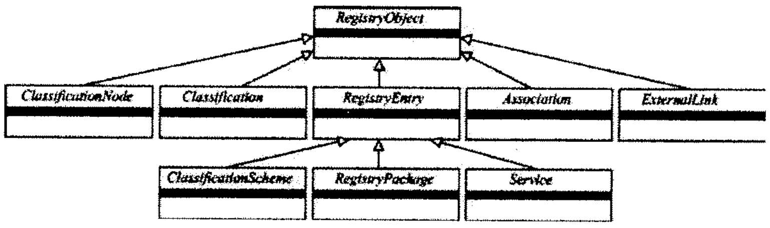


Figure 2. A part of the ebXML RIM class hierarchy.

registry and the repository via the ebXML registry API as defined in [16]. The API has two main interfaces:

- LifeCycleManager (LCM) is the interface responsible for all object lifecycle management requests.
- QueryManager (QM) is the interface responsible for handling all query requests.

The LifeCycleManager service enforces the life cycle rules for objects. The QueryManager interface of the ebXML Registry API provides access to the query service of the ebXML registry. A client uses the operations defined by this service to query the registry and discover objects. Supported query syntaxes include:

- An XML Filter Query syntax,
- An SQL-92 query, and
- A stored query syntax that allows client to invoke queries stored in the server by simply identifying the parameterized query and providing parameters for the query.

2.2.1. ebXML registry information model. The ebXML registry defines a Registry Information Model (RIM) [15] which specifies the standard metadata that may be submitted to the registry. This complements the ebXML Registry API which defines the interface clients may use to interact with the registry. Figure 2 presents the part of the ebXML RIM [15] related with storing metadata information. The main features of the information model include:

- *Registry object*: 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.
- *Object identification*: All RegistryObjects have a globally unique id, a human friendly name and a human friendly description.
- *Slot*: "Slot" instances provide a dynamic way to add arbitrary attributes to "RegistryObject" instances.
- *Object classification*: Any RegistryObject may be classified using ClassificationSchemes and ClassificationNodes which represent individual class hierarchy elements. A

Table 2. Predefined association types in ebXML registries.

Name	Description
Related To	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.

ClassificationScheme defines a tree structure made up of "ClassificationNodes". The ClassificationSchemes may be user-defined.

- *Object association*: Any RegistryObject may be associated with any other RegistryObject using an Association instance where one object is the sourceObject and the other is the targetObject of the Association instance. An Association instance may have an associationType which defines the nature of the association. There are a number of predefined *Association Types* that a registry must support to be ebXML compliant [15] as shown in Table 2. ebXML allows this list to be expanded.
- *Object organization*: RegistryObjects may be organized in a hierarchical structure using a familiar file and folder metaphor. The RegistryPackage instances serve as folders while RegistryObjects serve as files in this metaphor. In other words RegistryPackage instances group logically related RegistryObject instances together.
- *Service description*: The Service, ServiceBinding and SpecificationLink classes provide the ability to define service descriptions including WSDL and ebXML CPP/A.

As a summary, ebXML registry provides a persistent store for registry content. The current registry implementations store registry data in a relational database. ebXML Registry Services Specification defines a set of Registry Service interfaces which provide access to registry content. There are a set of methods that must be supported by each interface. A registry client program utilizes the services of the registry by invoking methods on one of these interfaces. The Query Manager component also uses these methods to construct the objects by obtaining the required data from the relational database through SQL queries. In other words, when a client submits a request to the registry, registry objects are constructed by retrieving the related information from the database through SQL queries and are served to the user through the methods of these objects.

3. Proposed enhancements to the ebXML registry architecture

Being OWL aware entails the following enhancements to the ebXML registry:

- *Representing OWL constructs through ebXML constructs*: ebXML provides a classification hierarchy made up of classification nodes and predefined type of associations between the registry objects. We represent *OWL Lite* constructs by using combinations of these constructs and define additional types of associations when necessary. For example, "OWL ObjectProperty" is defined by introducing a new association of type "objectProperty". The details of this work are presented in Section 4.
- *Automatically generating ebXML constructs from the OWL descriptions and storing the resulting constructs into the ebXML registry*: We developed a tool to create an ebXML Classification Hierarchy from a given OWL ontology automatically by using the transformations described in Section 4. The OWL file is parsed using Jena [24], 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.
- *Querying the registry for enhanced semantics*: We provide additional stored procedures to process the OWL semantics introduced in Section 4. A user can handle the OWL semantics by using these stored procedures or through SQL. Note that stored procedures and SQL are two of the supported query syntaxes in ebXML. In order to handle the OWL semantics transparently to the user through the third query syntax, namely, the "filter query", the Query Manager needs to be modified to invoke the related stored procedures we have introduced, when necessary.

The enhanced architecture is shown in figure 3. The OWL constructs are represented entirely through ebXML constructs by defining new types of associations which is allowed by the registry architecture. Hence there are no changes in the relational database schemas.

4. Providing OWL support to ebXML registries

In this section, we first describe how OWL constructs can be represented through ebXML registry information model constructs. We then provide the stored procedures to retrieve richer sets of results from the registry based on *OWL Lite* constructs. The stored procedures are defined using the ebXML relational schema specifications. The schemas used in the examples are given in Table 3.

Table 3. ebXML relational schemas.

ClassificationNode(accessControlPolicy, id, objectType, code, parent, path)
Association(accessControlPolicy, id, objectType, associationType, sourceObject, targetObject, isConfirmedBySourceOwner, isConfirmedByTargetOwner)
Name_(charset, lang, value, parent)

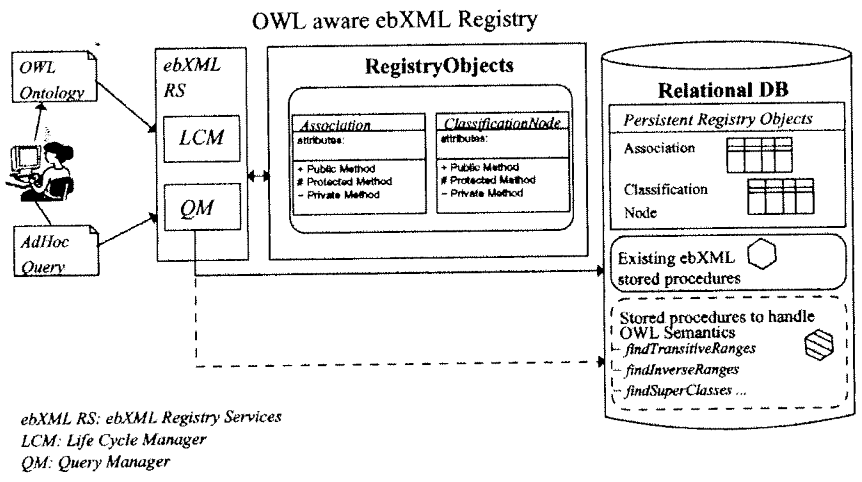


Figure 3. Enhancements to the ebXML registry architecture.

In Section 4.2, to demonstrate the benefits of the additional semantics incorporated into the ebXML registries, we describe a semantic-based service composition tool. This tool partially automates service discovery and composition in OWL aware ebXML registries.

4.1: *Mapping OWL ontologies through ebXML classification hierarchies and providing registry support for processing the OWL constructs*

From the descriptions presented in Section 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 provide the details of representing the *OWL Lite* constructs in an ebXML registry and then give the required stored procedures to process this semantics. Note that the mechanisms to represent currently available OWL Lite constructs can be thought of as a model for handling new OWL constructs that may appear or changes in existing constructs.

4.1.1. OWL classes and properties. 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 the target object of "rdfs:domain" property can be mapped to a "source RegistryObject" and the target object of "rdfs:range" can be mapped to a "target RegistryObject". In OWL, properties can be of two types:

- ObjectProperty type defines relations between instances of two classes.
- DatatypeProperty type defines relations between instances of classes and XML Schema datatypes.

To represent OWL ObjectProperty (or DatatypeProperty) in ebXML, we define a new type of association called "ObjectProperty" (or "DatatypeProperty"). Consider the following example which defines an object property "hasAirport" whose domain is "City" and whose range is "Airport":

```
<owl:ObjectProperty rdf:ID="hasAirport">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="#AirPort"/>
</owl:ObjectProperty>
```

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

```
<rim:ClassificationNode id='City' parent='Country'>
  </rim:ClassificationNode>
<rim:ClassificationNode id='Airport' parent='TravelThing'>
  </rim:ClassificationNode>
<rim:Association id='hasAirport' associationType='ObjectProperty'
  sourceObject = 'City' targetObject='Airport' >
</rim:Association>
```

Similarly, to represent OWL DatatypeProperty in ebXML, we define a new type of association called "DatatypeProperty". Consider the following example which defines an datatype property "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema nonNegativeInteger":

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

To describe this semantics, we define a new association of type "DatatypeProperty" as shown in the following:

```
<rim:Association id = 'hasPrice' associationType = 'DatatypeProperty'
   sourceObject = 'AirReservationServices'
   targetObject = 'integer' >
   <rim:Name> <rim:LocalizedString value ="hasPrice"/></rim:Name>
</rim:Association>
```

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, as demonstrated in the following:

```
<rim:ExternalLink id = "integer"
             externalURI="http://www.w3.org/2001/XMLSchema#integer">
      <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>
                 </rim:Name>
</rim:ExternalLink>
```

Once such ObjectProperty or DatatypeProperty definitions are stored in the ebXML registry, they can be retrieved through ebXML query facilities by the user. However, providing some stored procedures for this purpose facilitates the direct access. We therefore propose the following stored procedure to be available in the registry which retrieves all the object properties of a given classification node:

```
CREATE PROCEDURE findObjectProperties($className) AS
BEGIN
SELECT A.id
FROM Association A, Name_ N, ClassificationNode C
WHERE A.associationType LIKE 'objectProperty' AND
      C.id = N.parent AND
      N.value LIKE $className AND
      A.sourceObject = C.id
END;
```

A similar stored procedure can be given to retrieve datatype properties of a given class.

4.1.2. OWL class hierarchies. 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. In ebXML, Class Hierarchy is achieved by the "parent" association. However it is not possible to associate a ClassificationNode with two parents. In other words an ebXML Class hierarchy has a tree structure 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="AirReservationServices">
 <rdfs:subClassOf rdf:resource="http://www.daml.org/services/
    owl-s/1.0/Profile.owl#Profile"/>
 <rdfs:subClassOf  rdf:resource="#AirServices"/>
</owl:Class>
```

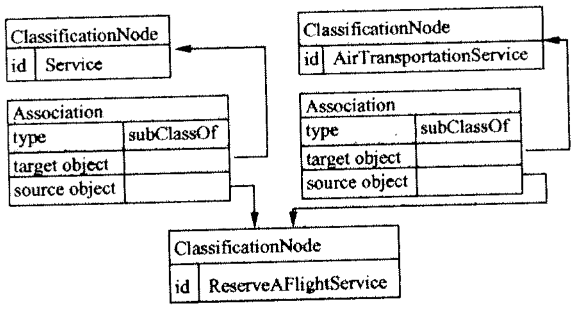


Figure 4. Representing an example "owl:subClassOf" property in ebXML registry.

Here, "AirReservationServices" service inherits both from "AirServices" service and OWL-S ServiceProfile class. Figure 4 shows how this is represented through ebXML RIM constructs. As presented in the figure, "AirReservationServices" ClassificationNode is associated with the "OWL-S Profile" and "AirServices" ClassificationNodes through the "target" and "source" object attributes of the newly created "subClassOf" ebXML Association.

Once we define such a semantics, we need the code to process the objects in the registry according to the semantics implied; that is, given a class, we should be able to retrieve all of its subclasses and/or all of its super classes. By making the required stored procedures available in the registry, this need can be readily served. For example, the following procedure finds all the immediate super classes of a given class:

```
CREATE PROCEDURE findSuperClasses($className) AS
BEGIN
SELECT C2.id
FROM Association A, Name_ N, ClassificationNode C1,
  ClassificationNode C2
WHERE A.associationType LIKE 'subClassOf' AND
      C1.id = N.parent AND
      N.value LIKE $className AND
      A.sourceObject = C1.id AND
      A.targetObject = C2.id
END;
```

Similar procedures can be provided to find all the superclasses of a given class (not only the immediate ones) as well as all its subclasses. The following procedure can then be used to retrieve all of the properties of a given class including the ones inherited from its super classes:

```
CREATE PROCEDURE findInheritedObjectProperties ($className) AS
SELECT A.id FROM Association A, ClassificationNode C WHERE
A.sourceObject=C.id AND
      A.associationType LIKE 'objectProperty' AND
      C.id IN (
            SELECT parent
            FROM Name_
            WHERE value LIKE $className
            UNION
            findSuperClasses($className)
            }
END;
```

4.1.3. OWL subPropertyOf. Since OWL properties are represented through ebXML associations, we define "rdfs:subPropertyOf" as an association between associations with a new association type of "subPropertyOf". The following procedure finds all the immediate super properties of a given property and similar procedures can be made available for all the super and subproperties:

```
CREATE PROCEDURE findSuperProperties($propertyName) AS
BEGIN
SELECT A3.id
FROM Association A1, Association A2, Association A3, Name_ N
WHERE A2.associationType LIKE 'subPropertyOf' AND
      A1.id = N.parent AND
      N.value LIKE $propertyName AND
      A2.sourceObject = A1.id AND
      A2.targetObject = A3.id
```

4.1.4. OWL equivalentClass, equivalentProperty and sameAs properties. In ebXML, the predefined "EquivalentTo" association (Table 2) expresses the fact that the source registry object is equivalent to target registry object. Therefore, "EquivalentTo" association is used to express "owl:equivalentClass", "equivalentProperty" and "sameAs" properties since classes, properties and instances are all ebXML registry objects.

Given a class, the following stored procedure retrieves all the equivalent classes:

```
CREATE PROCEDURE findEquivalentInstances($className) BEGIN SELECT
N.value FROM Service S, Name_ N WHERE S.id IN (
    SELECT classifiedObject
    FROM Classification
    WHERE classificationNode IN (
        SELECT id
        FROM ClassificationNode
        WHERE id IN (
            SELECT parent
            FROM Name_
            WHERE value LIKE $className
            )
        UNION
        SELECT A.targetObject
        FROM Association A, Name_ N, ClassificationNode C
        WHERE A.associationType LIKE 'EquivalentTo' AND
              C.id = N.parent AND
              N.value LIKE $className AND
              A.sourceObject = C.id
            )
    ) AND S.id=N.parent
END;
```

4.1.5. OWL transitive property. In OWL, if a property, P, is specified as transitive then for any x, y, and z: $P(x, y)$ and $P(y, z)$ implies $P(x, z)$. Transitive property can be defined as a new type of association in ebXML.

Consider the following example where we define the "succeeds" as a transitive property of "TravelWebService" class:

```
<owl:ObjectProperty rdf:ID="succeeds">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#TravelWebService"/>
  <rdfs:range rdf:resource="#TravelWebService"/>
</owl:ObjectProperty>
```

Assuming the following two definitions:

```
<TravelWebService rdf:ID="MyHotelAvailabilityService">
  <succeeds rdf:resource="#MyAirReservationService"/>
</TravelWebService>

<TravelWebService rdf:ID="MyInsuranceService">
  <succeeds rdf:resource="#MyHotelAvailabilityService"/>
</TravelWebService>
```

Since "succeeds" is a transitive property, it follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly stated.

To make any use of this transitive property in ebXML registries, coding is necesary to find out the related information. We provide the following stored procedure to handle this semantics: Given a class which is a source of a transitive property, this stored procedure retrieves not only the target of a given transitive property, but if the target objects have the same property, it also retrieves their target objects too.

```
CREATE PROCEDURE findTransitiveRelationships($className,
$propertyName) BEGIN SELECT A2.targetObject FROM Association A1,
Association A2, Name_ N1,Name_ N2, Name_ N3 WHERE
A1.associationType LIKE 'transitiveProperty' AND
      A1.id = N1.parent AND
      N1.value LIKE $propertyName AND
      A1.sourceObject = N3.parent AND
      N3.value LIKE $className AND
      A2.sourceObject = A1.targetObject AND
      A2.id = N2.parent AND
      N2.value LIKE $propertyName AND
      A2.associationType LIKE 'transitiveProperty'
UNION
SELECT A1.targetObject
FROM Association A1, Name_ N1, Name_ N3
WHERE A1.associationType LIKE 'transitiveProperty' AND
```

```
      A1.id = N1.parent AND
      N1.value LIKE $propertyName AND
      A1.sourceObject = N3.parent AND
      N3.value LIKE $className
END;
```

4.1.6. OWL inverseOf property. In OWL, if a property, $P1$, is tagged as the "owl:inverseOf" $P2$, then for all x and y: $P1(x, y)$ if $P2(y, x)$. Consider for example the "succeeds" property defined in Section 4.1.5. To denote that a certain Web service instance precedes another, we may define the "precedes" property as an inverse of the "succeeds" property as follows:

```
<owl:ObjectProperty rdf:ID="precedes">
  <owl:inverseOf rdf:resource="#succeeds"/>
</owl:ObjectProperty>
```

Then, by using the following stored procedure, we can find all the services that precede a given service by making use of its "succeeds" property.

```
CREATE PROCEDURE findInverseRanges($className, $propertyName)
BEGIN
SELECT C2.id
FROM Association A, Name_ N, Name_ N2, ClassificationNode C1,
      ClassificationNode C2
WHERE A.id=N2.parent AND
      N2.value LIKE $propertyName AND
      C1.id = N.parent AND
      N.value LIKE $className AND
      A.sourceObject = C1.id AND
      A.targetObject = C2.id
UNION
SELECT A3.sourceObject
FROM Association A1, Association A2, Association A3, Name_ N,
      NAME_ N2, ClassificationNode C1
WHERE A2.associationType LIKE 'inverseOf' AND
      A1.id = N.parent AND
      N.value LIKE $propertyName AND
      A2.sourceObject = A1.id AND
      A3.id=A2.targetObject AND
      C1.id = N2.parent AND
      N2.value LIKE $className AND
      A3.targetObject = C1.id
END;
```

4.1.7. OWL restriction. 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:Restriction", 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.

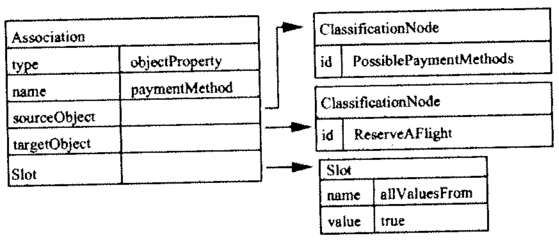


Figure 5. Representing an example OWL restriction in ebXML registry.

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="AirReservationServices">
  <rdfs:subClassOf rdf:resource="&service"/>
  <rdfs:subClassOf rdf:resource= "#AirServices"/>
  <rdfs:subClassOf>
    <owlRestriction>
    <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 "AirReservationServices" 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 already defined 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 5 shows how the example above is represented through ebXML RIM constructs. Through the "allValuesFrom" slot added to the "paymentMethod" association, the type of the restriction is stated explicitly.

4.1.8. OWL class intersection. OWL provides the means to manipulate class extensions using basic set operators. In *OWL Lite*, only "owl:intersectionOf" is available which defines a class that consists of exactly all objects that belong to both of the classes. Consider the following example:

```
<owl:Class rdf:ID="AirReservationServices">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#AirServices"/>
    <owl:Class rdf:about="#ReservationServices"/>
  </owl:intersectionOf>
</owl:Class>
```

In ebXML RIM "owl:intersectionOf" set operator can be expressed as follows:

- A new association type called "intersectionOf" is created.
- The classes constituting the intersection are represented as members of a Registry Package.
- The source object of the set operator is assigned as the sourceObject of the "intersectionOf" association.
- The target object of the "intersectionOf" association is set to be the newly created RegistryPackage.

The RIM representation of the OWL example presented above is presented in [12]. When such a representation is used to create a complex class in RIM, it becomes possible to infer that the objects classified by both of the classes constituting the intersection are also the instances of this complex class. The stored procedure retrieves the direct instances of the complex class and also the intersection of the instances of the member classes can be found in [12].

Table 4 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.

4.2. *How to exploit OWL semantics for service discovery and composition in ebXML registry*

In this section, in order to demonstrate the benefits of the proposed enhancements to the ebXML registry, we describe discovery of semantically enriched Web services through a "Web Service Discovery and Composition Definition Tool". This tool aids the user to find appropriate Web Services through automated service discovery in OWL aware ebXML registries and compose them into a workflow. Note that the emphasis of the work described is on semantic discovery of Web services not their semantic composition.

As presented in figure 6, the tool allows the ebXML classification hierarchies to be depicted graphically. When a user clicks on a node in the classification hierarchy, the generic properties of the service are retrieved from the registry and revealed to the user as depicted in figure 9. The user can fill in the desired properties of services she is looking for through this GUI. The tool queries the ebXML registry automatically to find the services that satisfy user constraints and the results are presented to the user in the second pane of figure 6. Then, a user can select from the presented service instances, can place the service to the appropriate slot in service choreography by dragging and dropping, and continues composition definition by adding control blocks when necessary. When the composition is finalized the BPEL4WS definition is created based on the WSDL files of the Web service instances retrieved from the ebXML registry.

Table 4. Mapping OWL ontologies to ebXML classification hierarchies without affecting the registry.

OWL	ebXML
owl:Class	ClassificationNode
individual	RegistryObject
rdf:Property	Association
rdfs:domain	sourceObject
rdfs:range	targetObject
owl:equivalentTo owl:samePropertyAs owl:sameAs	An association with a predefined association type of "EquivalentTo".
owl:differentFrom	An new association type of "differentFrom" is defined.
owl:AllDifferent owl:distinctMembers	A new association type "distinctMembers" is defined to add members to a Registry Package.
An association with a new association type is defined.	
rdfs:subClassOf owl:ObjectProperty owl:disjointWith owl:TransitiveProperty owl:FunctionalProperty owl:InverseFunctionalProperty owl:SymetricProperty	"subClassOf" "objectProperty" "disjointWith" "transitiveProperty" "functionalProperty" "inverseFunctionalProperty" "symetricProperty"
owl:DataTypeProperty	XML Schema datatypes are used by providing an external link from the registry.
rdfs:subPropertyOf owl:inverseOf	An association between associations with a new association type "subPropertyOf"/"inverseOf" is defined.
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:Restriction	Since ebXML RIM associations have local scope, only the type of the Restriction needs to be specified.
A slot type is defined for the association representing a restriction.	
owl:allValuesFrom owl:someValuesFrom owl:hasValue	"allValuesFrom" "someValuesFrom" "hasValue"
An association with a new association type is defined.	
owl:cardinality owl:minCardinality owl:maxCardinality	"cardinality" "minCardinality" "maxCardinality"

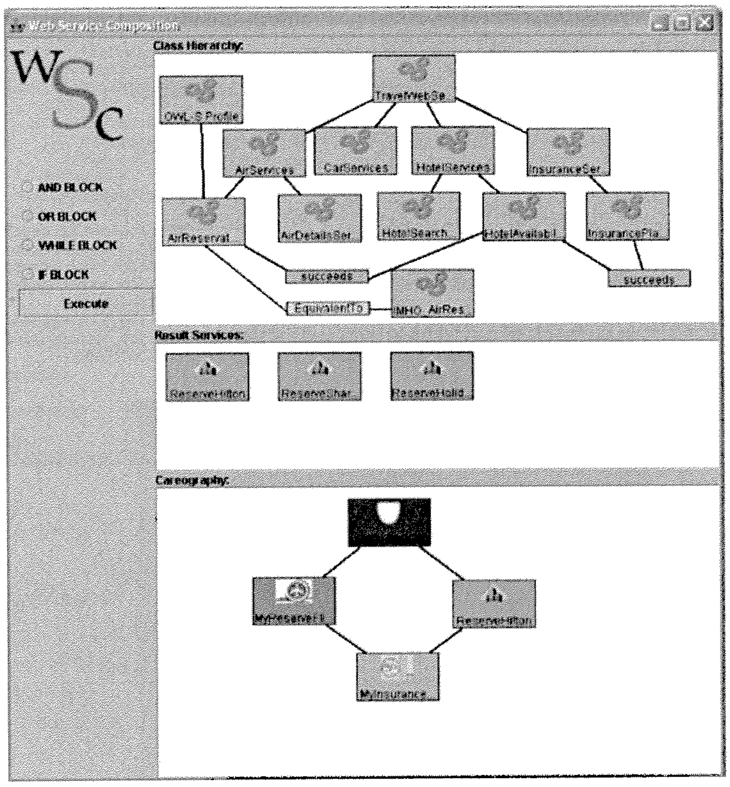


Figure 6. A snapshot of the GUI tool for semantics-based web service composition for ebXML registries.

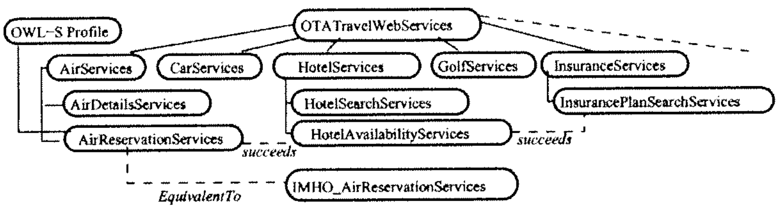


Figure 7. An example travel ontology.

In the following, we present an example to clarify how this semantic discovery mechanism works. Assume the travel ontology given in figure 7 is stored in the ebXML registry and is used in annotating travel Web services. Note that this ontology is based on "Open Travel Alliance" (OTA) specifications [30]. Assume further that a user wishes to organize a trip by first reserving a flight. By selecting the "AirReservationServices" ebXML ClassificationNode presented in the Web Service Composition tool, it becomes possible to query

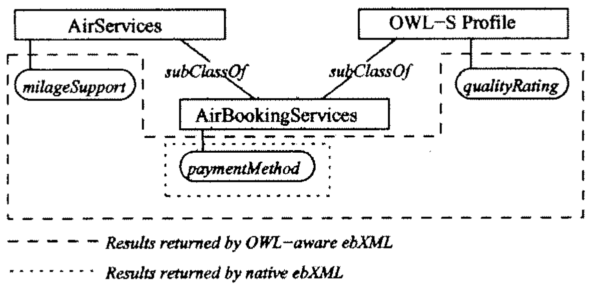
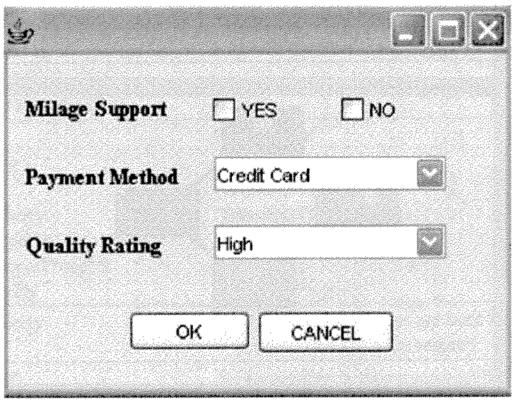


Figure 8. ebXML semantic support for class hierarchies.



Milage Support [ ] YES [ ] NO

Payment Method: Credit Card

Quality Rating: High

Figure 9. A GUI tool to obtain the service property values from the user.

the services that are classified under the generic "AirReservationServices" node. In doing this, it is necessary to retrieve the properties of this class so that the user can provide her preferred values for the properties.

As presented in figure 7, "AirReservationServices" are defined as a subclass of both "OWL-S Profile" class and the "AirServices" class. In conventional ebXML, when a user submits a query to the ebXML registry to get the object properties of the "AirReservation-Services", only the immediate associations that are of type "objectProperty" are returned as presented in figure 8. However by exploiting the semantic capabilities of the OWL aware ebXML, the user can call the stored procedure "findInheritedObjectProperties" defined in Section 4.1.2 to retrieve the properties inherited from the parent classes too (figure 8).

These properties are shown to the user through the GUI depicted in figure 9. Once the user provides the preferred values, the instances satisfying these values are retrieved through the ebXML Filter query shown in figure 10 which is automatically issued through the tool. Note that while storing the Web service instances to ebXML registries, the values of their properties are represented through "slot" values.

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

Figure 10. An example filter query for retrieving the instances of the "AirReservationService".

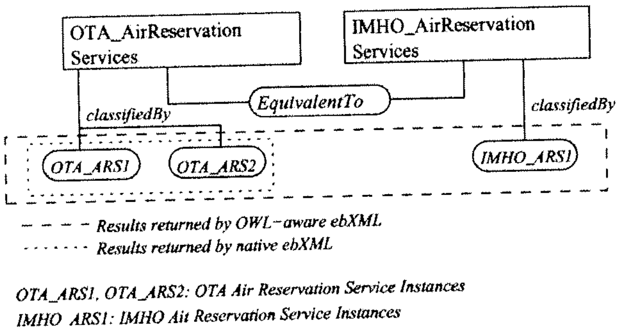

OTA_AirReservation Services
IMHO_AirReservation Services
EquivalentTo
classifiedBy
classifiedBy
OTA_ARS1
OTA_ARS2
IMHO_ARS1
– – – – Results returned by OWL-aware ebXML
· · · · · · · Results returned by native ebXML
OTA_ARS1, OTA_ARS2: OTA Air Reservation Service Instances
IMHO ARS1: IMHO Ait Reservation Service Instances

Figure 11. ebXML semantic support for equivalent classes.

Note further that the AirReservationServices node in the OTA Travel ontology (figure 7) is defined to be equivalent to "IMHO_AirReservationServices" (IMHO stands for "Interoperable Minimum Harmonise Ontology" which is a tourism ontology [22]). This relation is represented through the "EquivalentTo" type association of ebXML.

Without OWL semantic support, when the Filter Query presented in figure 10 is issued, the ebXML Query Manager will retrieve the services classified only by the AirReservationServices as presented in figure 11, using the SQL query presented in figure 12.

With OWL semantic support, our tool processes the semantics of the "EquivalentTo" property to retrieve the instances of the AirReservationServices by using the "findEquivalentInstances(AirReservationServices)" stored procedure defined in Section 4.1.4. Note that the use of these stored procedures is not restricted to our tool; any ebXML client can also use these stored procedures.

Assuming that the user chooses the "MyAirReservationService" instance among the Web services presented to her and discovering that 'AirReservationServices" has a "succeeds"

```
SELECT *  FROM Service WHERE id IN (
    SELECT classifiedObject FROM Classification
    WHERE classificationNode IN (
        SELECT id FROM ClassificationNode
        WHERE id IN (
            SELECT parent FROM name_
            WHERE value='AirReservationServices'
        )
    )
) AND id IN (
    SELECT parent FROM Slot
    WHERE name_='paymentMethod' AND
    value LIKE '%CreditCard%')
```

Figure 12. SQL query to retrieve the services classifed with "AirReservationServices".

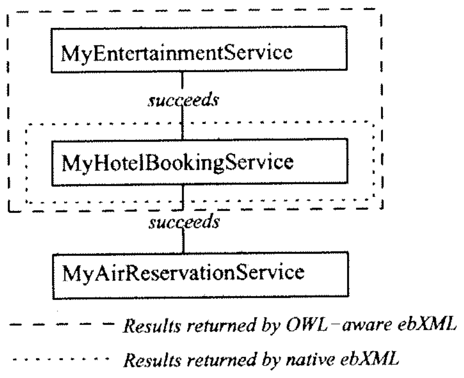

MyEntertainmentService
succeeds
MyHotelBookingService
succeeds
MyAirReservationService
– – – – – Results returned by OWL-aware ebXML
· · · · · · · · · Results returned by native ebXML

Figure 13. ebXML semantic support for transitive properties.

property, she may wish to consult to the ebXML for finding the "succeeding" services of this instance. Consider the example given in Section 4.1.5. When a user wishes to retrieve the "succeeding" services of the "MyAirReservationService" instance and issues a query to the ebXML registry, without OWL semantic support only "MyHotelAvailabilityService" instance will be returned as presented in figure 13, although "succeeds" has been declared to be transitive.

To be able to exploit the "transitivity" semantics, our tool uses the "findTransitiveRelationships(AirReservationServices,succeeds)" stored procedure defined in Section 4.1.5, which returns the "MyInsuranceService" instance additionally. Then the user can add these service instances to her choreography and to obtain the BPEL4WS definition.

4.3. *Implementation status*

A proof of concept implementation of the system is realized by using OASIS ebXML Registry Reference Implementation [18]. As an application server to host Web services to be accessed through SOAP, Apache Tomcat 4.1 [43] is used. The WSDL descriptions of the implemented services are generated through IBM Web Services Toolkit 3.2 (WSTK) [21], and BPEL4WS definitions are generated and executed using BPWS4J [20]. Finally OWL ontology is parsed with Jena 2.1 OWL parser [24].

5. Related work

In the early nineties, ontologies have been a research topic being addressed in a rather small research community. This has changed drastically in the late nineties by the insight that a conceptual, yet executable model of an application domain provides a significant value [17, 42]. The impact has increased with the Semantic Web initiative and the Web Ontology Language (OWL) [31].

The importance of semantics is also recognized in the Web services area and there have been several efforts to improve the semantics support for Web services [3, 4].

The need for extending the UDDI [44] registries with semantic capabilities has been addressed in the literature [8, 9, 34]. Note that UDDI registries use tModels to represent compliance with a taxonomy such as Universal Standard Products and Services Classification [46]. [8, 9] describe a mechanism to relate DAML-S ontologies with services advertised in the UDDI registries. [34] also addresses importing semantic to UDDI registries where DAML-S specific attributes such as *inputs, outputs* and *geographicRadius* are represented using tModel mechanisms of UDDI. An extended UDDI registry is also reported in [39] which allows to record user defined properties associated with a service and then to enable discovery of services based on these. In [40], the authors discuss adding semantics to WSDL using DAML+OIL ontologies. Their approach also uses UDDI to store these semantic annotations and search for Web services based on them.

As presented, although there has been a considerable amount of research for extending UDDI registries with OWL-S semantics for facilitating the discovery of Web Services, ebXML registries has not been studied much in this context. 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.

Related with ebXML, exploiting the native class hierarchies in ebXML registries for service discovery and composition is described in [10]. In [11], we present some initial ideas about enriching ebXML registries with OWL semantics. The work presented in this paper extends the ideas given in [11] and provides a complete model on how *OWL Lite* constructs can be stored to ebXML registries and queried to facilitate semantic discovery of Web Services.

An important effort in defining the semantics of Web Services is OWL-S [32] (previously DAML-S) which defined an upper ontology to describe service semantics. Related with exploiting DAML-S for service discovery and composition, some of the previous work use AI techniques to match the inputs and outputs of services requested and advertised. For example, [33] describes a matching engine to match advertised services with service requests, both defined in DAML-S. In [33], an advertisement matches a request when all the outputs of the request are matched by the outputs of the advertisement, and all the inputs of the advertisement are matched by the inputs of the request. In [47], DAML-S is extended to describe bioinformatics Web services and the services are matched by subsumption reasoning over the service descriptions.

In contrast to the approaches described above, we take a data management approach for service discovery part of the system by exploiting the metadata and the query mechanism of the ebXML registry. In our work, rather than having a user specifying the inputs and outputs of a request, the ebXML registry is queried through the enhanced registry constructs to obtain the semantic information about the Web services. In this way service discovery reduces to querying the registry with the help of the ontology.

Medjahed et al. [26] proposes an ontology-based framework for the automatic composition of Web services. The authors present a technique to generate composite services from high-level declarative descriptions called "Composite Service Specification Language" (CSSL). For describing the semantics of Web Services, they extend WSDL with semantic capabilities. Composition plans are generated through a matchmaking algorithm according to composer's specifications in CSSL. Matchmaking algorithm takes a UDDI registry hosting Web Service WSDL definitions (extended with semantic constructs), retrieves a set of services from UDDI registry through the "service category" defined in CSSL, decides the semantic and syntactic "compasability" of Web services by comparing each Web service in this set with the preceding service in CSSL definition.

While this paper extends WSDL definitions with semantic constructs, in our approach domain ontologies are stored in ebXML registries, and semantics of Web services are defined through associating them to the ontology nodes in the registry. Furthermore, the work presented in [26] concentrates on the matchmaking of semantically enriched WSDL files and complements by our approach as follows: we show in detail how the semantic mechanisms of ebXML registries can be used to discover Web services semantically through registry queries. This phase provides the set of semantically suitable Web services prunning the unrelated services. After this step, a matchmaking algorithm as presented in [26] can be used to check their syntactic "composability".

Cardoso and Sheth [5] describes an algorithm to discover Web services and resolve heterogeneity among their interfaces and the workflow host. In this architecture, service discovery is achieved as follows: users can advertise the DAML-S definitions of their Web services to a registry, which is a service capability table where service descriptions are added, through a registry service. Through a discovery service, DAML-S profiles representing template of the queried service is sent to the system, then through a matchmaking algorithm presented in the paper, the services matched are presented to the user based on specific ranking criteria.

Our work complements this approach as follows: [5] proposes a generic semantic service discovery mechanism without addressing what a specific service registry may offer to help with service discovery. In our work, we show how to exploit the semantic constructs and query facilities of a specific service registry, namely, ebXML which is a widely adopted industry standard.

OASIS open source standards body [27] also has a number of standardization efforts aiming to support semantics within ebXML Framework. Some of these efforts will be presented in this section to put the work described in this paper into perspective. Note that the work described in this paper is also presented as a proposal to OASIS ebXML Semantic Content Management subcommittee which is working on possible semantic extensions to the registry.

There are several other key semantic requirements being addressed within the OASIS open source standards body [27]. This work is progressing through the committees including: The Business-Centric Methodology (BCM) Technical Committee (TC) [28], The ebXML Registry Semantic Content Management Sub-Committee (ebXMLR-SCM) [29]:

- *The business-centric methodology (BCM) technical committee (TC)* [28]: BCM addresses a proper interpretation of the business language semantics found in a SOA (Service Oriented Architecture) metadata framework/classification system which is essential for harnessing tacit knowledge and facilitating shared communications.
- *The ebXMLregistry semantic content management sub-committee (ebXMLR-SCM)* [29]: A key factor of the ebXMLR-SCM work towards semantic extensions of the Registry/Repository is the acknowledgment that the mapping of e-business artifacts to a semantic structure can employ many types of registry objects.

6. Conclusions and future work

This paper describes an engineering effort on how an ebXML registry can be made OWL aware. The work presented provides the foundation for OWL ontologies to be expressed in the registry. The representation of OWL semantics directly in the RIM enables the standard ebXML query facility to use stored procedures that can return a richer set of results based on explicit OWL constructs. As demonstrated, the queries of this type provide new capabilities to the ebXML client applications.

In this work, we use OWL Lite, since we are using ontologies to get knowledge through querying rather than reasoning. We investigate the possible ways of making the registry OWL aware and describe an approach that minimizes the changes on the ebXML specification.

There are two observations resulting from this experience:

- Ontologies can play two major roles: one is to provide a source of shared and precisely defined terms which can be used formalizing knowledge and relationship among objects in a domain of interest. 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, when we want to infer new information from the existing knowledge, we need reasoners. And reasoners can not directly run on the ebXML registry because all the registry information is stored in relational databases. Hence, there is a need to reconstruct the ontology from its representation in the ebXML registry.
- An ebXML registry client can use stored procedures that we have introduced to handle the OWL semantics. However, handling this semantics through the filter query in a transparent way to the user requires some modifications in the Query Manager Component of the registry. ebXML filter query, is designed to retrieve the registry objects as specified in the original RIM. It falls short to retrieve additional semantics introduced in this work.

  In "filter query", the user expresses what is to be retrieved from the registry as an XML message and the current syntax of ebXML query uses the conventional ebXML registry constructs. In order to retrieve extended semantics from in an OWL aware ebXML registry, through a "filter query", the Query Manager component needs to be extended.

  Consider the example defined in Section 4.1.8, where "AirReservationServices" is defined to be the intersection of the classes "AirServices" and "ReservationServices". When a user sends a Filter query to retrieve services classified by the "AirReservationServices" node, normally the ebXML Query Manager will return the services directly classified

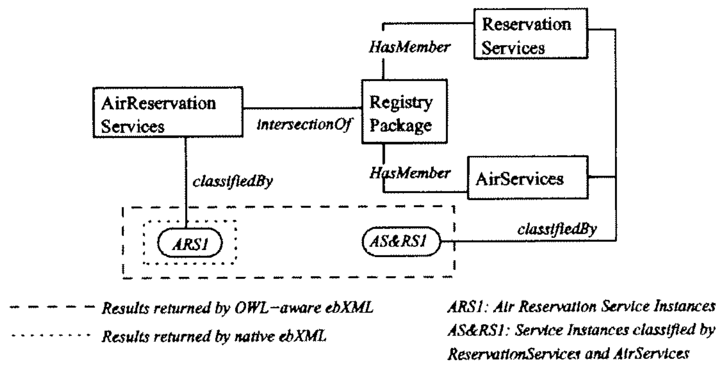

Reservation Services
HasMember
AirReservation Services
IntersectionOf
Registry Package
HasMember
AirServices
classifiedBy
ARS1
AS&RS1
classifiedBy
Results returned by OWL-aware ebXML
Results returned by native ebXML
ARS1: Air Reservation Service Instances
AS&RS1: Service Instances classified by ReservationServices and AirServices

Figure 14. ebXML semantic support for class intersection.

by "AirReservationServices" node. However with OWL support it is possible to retrieve the services classified by both of the "AirServices" and "ReservationServices" at the same time, and thus retrieving the instances of "AirReservationServices", as presented in figure 14.

To handle such a semantics, the ebXML Query Manager should be updated to execute the "findInstances($className)" stored procedure defined in Section 4.1.8 whenever it receives such a filter query. In fact, the Query Manager needs to consider all such possibilities and this can only be handled through reasoning.

There are a number of public domain and commercial OWL reasoners such as [6, 13, 36]. As a future work, we intend to improve the Query Manager component with reasoning capabilities by exploiting one of the existing OWL reasoners.

References

1. G. Antoniou and F. Harmalen, Web ontology language: OWL, Handbook on Ontologies, Springer, 2004.
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific American, May 2001.
3. C. Bussler, D. Fensel, T. Payne, and K. Sycara, "ISWC tutorial: semantic web services," http://www.daml.-ri.cmu.edu/tutorial/iswc-t3.html#2b
4. C. Bussler, D. Fensel, and A. Maedche, "A conceptual architecture for semantic web enabled web services," ACM Sigmod Record, vol. 31, no. 4, December 2002.
5. J. Cardoso and A. Sheth, "Semantic e-workflow composition," Journal of Intelligent Information Systems (JIIS), vol. 12, no. 3, 2003.
6. Cerebra OWL Reasoner, http://www.networkinference.com/Products/Cerebra_Server.html
7. DAML+OIL Reference Description, W3C Note, http://www.w3.org/2001/10/daml+oil, December 2001.
8. A. Dogac, I. Cingil, G.B. Laleci, and Y. Kabak, "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.
9. A. Dogac, G. Laleci, Y. Kabak, and I. Cingil, "Exploiting web service semantics: Taxonomies vs. ontologies," IEEE Data Engineering Bulletin, vol. 25, no. 4, December 2002.
10. A. Dogac, Y. Kabak, and G. Laleci, "A semantic-based web service composition facility for ebXML registries," in 9th International Conference of Concurrent Enterprising, Espoo, Finland, June 2003.

11. A. Dogac, Y. Kabak, and G. Laleci, "Enriching ebXML registries with OWL ontologies for efficient service discovery," in Proc. of RIDE'04, Boston, March 2004.
12. Example RIM representation of an Intersection, http://www.srdc.metu.edu.tr/~banu/dapd/-RIMIntersectionExample Example Stored Procedure, http://www.srdc.metu.edu.tr/~banu/dapd -/findInstancesStoredProcedure;
13. F-OWL Reasoner, http://fowl.sourceforge.net
14. ebXML, http://www.ebxml.org/
15. ebXML Registry Information Model v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/-specs/ebRIM.pdf
16. ebXML Registry Services Specification v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/-specs/ebRIM.pdf
17. D. Fensel, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.
18. freebXML Registry Open Source Project http://ebxmlrr.sourceforge.net
19. I. Horrocks, "DAML+OIL: A description logic for the semantic web," IEEE Data Engineering Bulletin, vol. 25, no. 1, March 2002.
20. IBM Business Process Execution Language for Web Services Java, http://www.alphaworks.ibm.com/tech/-bpws4j
21. IBM Web Services Toolkit (WSTK), http://alphaworks.ibm.com/tech/webservicestoolkit.
22. Minimum Harmonise Ontology, http://www.harmo-ten.info/index. -php?option= content&task= view&id-=5&Itemid=29
23. IST-2104-SATINE Project, http://www.srdc.metu.edu.tr/webpage/projects/satine/
24. Jena2 Semantic Web Toolkit, http://www.hpl.hp.com/semweb/jena2.htm
25. D. McGuinness and F. Harmelen, "OWL web ontology language overview," W3C Recommendation, February 2004, http://www.w3.org/TR/owl-features/
26. B. Medjahed, A. Bouguettaya, and A. Elmagarmid, "Composing web services on the semantic web," VLDB Journal, vol. 12, no. 4, 2003.
27. OASIS, http://www.oasis-open.org/home/index.php
28. OASIS Business-Centric Methodology (BCM) Technical Committee (TC), http://www.oasis-open.org/-committees/tc_home.php?wg_abbrev=bcm
29. OASIS ebXML Registry Semantic Content Management SC, http://www.oasis-open.org/apps/-org/workgroup/regrep-semantic/
30. Open Travel Alliance, http://www.opentravel.org/
31. OWL Web Ontology Language 1.0 Reference http://www.w3.org/TR/2002/WD-owl-ref-20020729/ref-daml
32. OWL-S, http://www.daml.org/services/daml-s/0.9/
33. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in Proc. of Intl. Semantic Web Conference, Sardinia, Italy, June 2002.
34. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Importing the semantic web in UDDI," in Web Services, E-Business and Semantic Web Workshop, 2002.
35. Registering web services in an ebXML Registry version 1.0. http://www.oasis-open.org/apps/org/workgroup/-regrep/download.php/1636/OASIS-Registry
36. Pellet OWL Reasoner, http://www.mindswap.org/2003/pellet/
37. RDF Schema: Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.
38. RDF Syntax: Resource Description Framework Model and Syntax Specification, W3C Recommendation, 1999, http://www.w3.org/TR/REC-rdf-syntax.
39. A. ShaikhAli, O. Rana, R. Al-Ali, and D. Walker, "UDDIe: An extended registry for web services," Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT Conference, Florida, January 2003.
40. K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, "Adding semantics to web services standards," In proc. of ICWS, 2003.
41. M. Smith, C. Welty, and D. McGuinnes, "OWL web ontology language guide," W3C Recommendation, February 2004, http://www.w3.org/TR/owl-guide/

42. S. Staab and R. Studer, Handbook on Ontologies, Springer, 2004.
43. Tomcat, http://jakarta.apache.org/.
44. Universal Description, Discovery and Integration (UDDI), www.uddi.org
45. UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/-cefact/index.htm.
46. Universal Standard Products and Services Classification (UNSPSC) http://eccma.org/unspsc
47. C. Wroe, R. Stevens, C. Goble, and A. Roberts, Greenwood, M., "A suite of DAML+OIL ontologies to describe bioinformatics web services and data," Intl. Journal of Cooperative Information Systems, to appear.

Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain [*]

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

ABSTRACT

One of the most challenging problems in the healthcare domain is providing interoperability among healthcare information systems. In order to address this problem, we propose the semantic mediation of exchanged messages. Given that most of the messages exchanged in the healthcare domain are in EDI (Electronic Data Interchange) or XML format, we describe how to transform these messages into OWL (Web Ontology Language) ontology instances. The OWL message instances are then mediated through an ontology mapping tool that we developed, namely, OWLmt. OWLmt uses OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the mapping patterns defined through a GUI.

Through a prototype implementation, we demonstrate how to mediate between HL7 Version 2 and HL7 Version 3 messages. However, the framework proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use.

1. INTRODUCTION

Most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. A number of standardization efforts are progressing to address this interoperability problem such as EHRcom [5], openEHR [18] and HL7 Version 3 [8]. Yet, it is not realistic to expect all the healthcare institutes to conform to a single standard. Furthermore, different versions of the same standard (such as HL7 Version 2 and Version 3) and even the different implementations of the same standard, for example, some HL7 Version 2 implementations, do not interoperate. Therefore there is a need to address the interoperability problem at the semantic level. Semantic interoperability is the ability for information shared by systems to be understood at the level of formally defined domain concepts so that the information is computer processable by the receiving system [12].

In this paper, we describe an engineering approach developed within the scope of the Artemis project [1] to provide the exchange of meaningful clinical information among healthcare institutes through semantic mediation. The proposed framework, called AMEF (Artemis Message Exchange Framework) involves first providing the mapping of a source ontology into a target ontology with the help of a mapping tool which produces a mapping definition. This mapping definition is then used to automatically transform the source ontology message instances into target message instances.

Through a prototype implementation, we demonstrate how to mediate between HL7 Version 2 and HL7 Version 3 messages. However, the framework proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use.

The semantic mediation between HL7 Version 2 and HL7 Version 3 messages is realized in two phases:

- *Message Ontology Mapping Process*: In the first phase, the message ontologies of two healthcare institutes are mapped one another (Figure 1). Assume that healthcare institute A uses HL7 v2 and healthcare institute B uses HL7 v3 to provide system interconnection. The message ontologies of these institutes are mapped one into other by using an ontology mapping tool. For this purpose we have developed an OWL (Web Ontology Language) ontology mapping tool, namely, OWLmt [20]. With the help of a GUI, OWLmt allows to define semantic mappings between structurally different but semantically overlapping OWL ontologies, and produces a "Mapping Definition".

  Since message ontologies for HL7 messages do not exist yet, we use the HL7 Version 2 and Version 3 XML Schemas (XSDs) [27] to generate OWL ontologies. This process, called "Conceptual Normalization" [7] produces a "Normalization map" describing how a specific message XSD is transformed into the corresponding OWL schema.

  The "Mapping Definitions" and the "Normalization map" produced in the first phase are used during the second phase to automatically transform the message

*This work is supported by the European Commission through IST-1-002103-STP Artemis project and in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 104E013

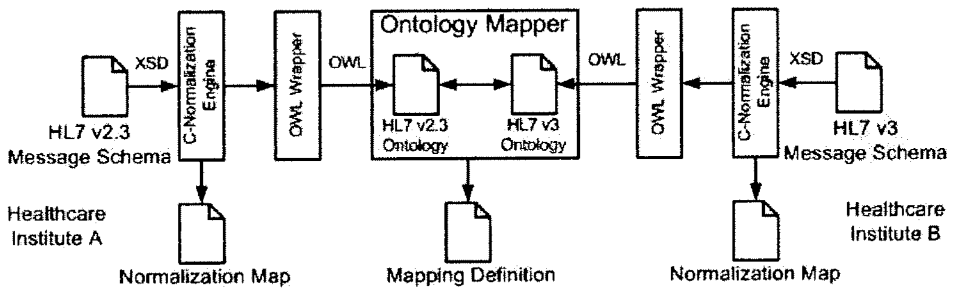


Figure 1: Message Schema Mapping Process

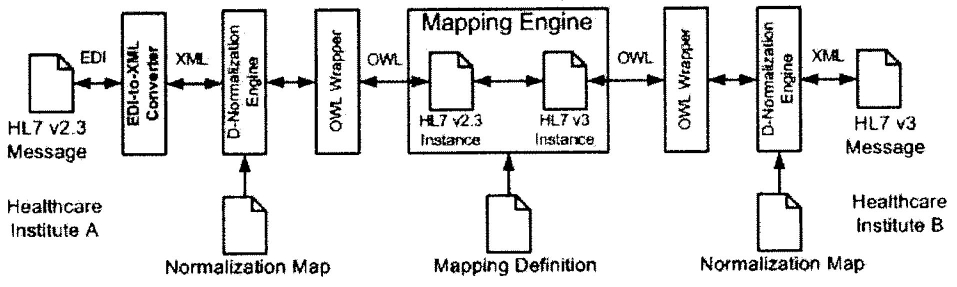


Figure 2: Automatic Message Instance Transformation Process

instances one into another.

- *Message Instance Mapping*: In the second phase (Figure 2), first the XML message instances of healthcare institute A are transformed into OWL instances by using the "Data Normalization" engine [7]. Note that if the message is in EDI (Electronic Data Interchange) format, it is first converted to XML. Then by using the *Mapping definitions*, OWL source (healthcare institute A) messages instances are transformed into the OWL target (healthcare institute B) message instances. Finally the OWL messages are converted to XML again through the "Data Normalization" engine.

The paper is organized as follows: In Section 2, we briefly summarize the HL7 standard. Section 3 describes the semantic mediation of HL7 v2 and v3 messages. The details of OWL mapping tool used in the mediation is presented in Section 4. Transforming HL7 v2 EDI messages to XML is briefly introduced in Section 5. Section 6 describes the "Normalization" tool used and the improvements realised on this tool. Finally, Section 8 concludes the paper.

2. HEALTH LEVEL 7 (HL7) STANDARD

The primary goal of HL7 is to provide standards for the exchange of data among healthcare computer applications. The standard is developed with the assumption that an event in the healthcare world, called the *trigger event*, causes exchange of messages between a pair of applications. When an event occurs in an HL7 compliant system, an HL7 message is prepared by collecting the necessary data from the underlying systems and it is passed to the requestor, usually as an EDI message. For example, as a result of a trigger event, say "I05", the clinical patient information for a given

RQC Request Clinical Information	
MSH	Message Header
QRD	Query Definition
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ NK1 }]	Next of Kin/Associated Parties
[{ GT1 }]	Guarantor
[{ NTE }]	Notes and Comments

RCI Return Clinical Information	
MSH	Message Header
MSA	Message Acknowledgment
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ DG1 }]	Diagnosis
[{ DRG }]	Diagnosis Related Group
[{ AL1 }]	Allergy Information
[	
{	
OBR	Observation Request
[{ NTE }]	Notes and Comments
[	
{	
OBX	Observation Result
[{ NTE }]	Notes and Comments
}	
]	
}	
]	
[{ NTE }]	Notes and Comments

Figure 3: The Structures of the RQC/RCI EDI messages for the HL7 Version 2 event I05

patient identifier is passed to the requestor as shown in Figure 3. Clinical information refers to the data contained in a patient record such as problem lists, lab results, current medications, family history, etc. [9].

HL7 version 2 is the most widely implemented healthcare informatics standard in the world today. Yet being HL7 Version 2 compliant does not imply direct interoperability between healthcare systems. Version 2 messages, contain many optional data fields. For example every attribute presented in square brackets in Figure 3, denotes optional information that may be omitted. This optionality provides great flexibility, but necessitates detailed bilateral agreements among

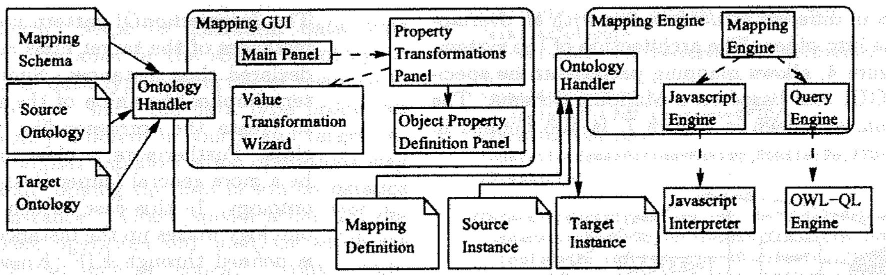


Figure 4: Architecture of OWLmt

the healthcare systems to achieve interoperability.

To remedy this problem, HL7 has developed Version 3 [8] which is based on an object-oriented data model, called Reference Information Model (RIM) [10]. The main objective of the HL7 Version 3 is to eliminate the optionality. RIM is used as the source of the content of messages and this results in a more efficient message development process. The result of the Version 3 process is the Hierarchical Message Definition (HMD), which defines the schema of the messages based on the RIM classes. Note that HL7 Version 3 messages do not interoperate with HL7 Version 2 messages.

3. SEMANTIC MEDIATION OF HL7 V2 AND V3 MESSAGES

In Artemis Message Exchange Framework (AMEF), the semantic mediation of HL7 v2 and v3 messages is realized in two phases:

- *Message Schema Mapping Process*: In the first phase, the message schemas of two healthcare institutes are mapped one another through semantic mediation as shown in Figure 1. At the heart of this process is the OWL Mapping tool, OWLmt, transforming OWL ontologies one into other.

  The OWL ontologies corresponding to the message schemas involved are generated through a set of available tools. First, for healthcare institute A (Figure 1), the HL7 Version 2 XML Schemas (XSDs) [27] are converted to RDFS (Resource Description Framework Schema) by using the Conceptual Normalization (C-Normalization) engine of the Harmonise project [7]. This process uses a set of heuristics as described in Section 6 and produces a "Normalization map" describing how a specific HL7 Version 2 message XSD is transformed into the corresponding RDFS schema and vice-versa. Then, by using the OWL Wrapper, which we developed using Jena API [13], RDFS Schemas are transformed to OWL.

  On the other hand, for healthcare institute B (Figure 1), in order to generate the XSDs of HL7 v3 messages, RoseTree tool of HL7 is used [24]. RoseTree allows the user to graphically build a HMD (Hierarchical Message Definition) from the Reference Information Model of HL7 v3. This generated HMD file describes the structure of the v3 XML messages, but it is not in XSD format. In order to translate the HMD file to XSD, "HL7 v3 Schema Generator" [11] is used.

  The next step is to map the source ontology into the target ontology by using OWLmt. This process is described in detail in Section 4.

- *Message Instance Mapping*: In the second phase (Figure 2), first the HL7 version 2 EDI messages are converted to XML. The open-source programming library from HL7, namely, HL7 application programming interface (HAPI) [6] is used for transforming the EDI messages into their XML representations.

  In the next step, as shown in Figure 2, the XML message instances of healthcare institute A are transformed to OWL instances by the "Data Normalization (D-Normalization) engine [7] using the "Normalization map" produced during the first phase.

  Then by using the *Mapping definitions*, OWLmt transforms OWL source (healthcare institute A) messages instances into the OWL target (healthcare institute B) message instances. Finally the OWL messages are converted to the XML format that the healthcare institute B understands, again through the "Data Normalization" engine as shown in Figure 2.

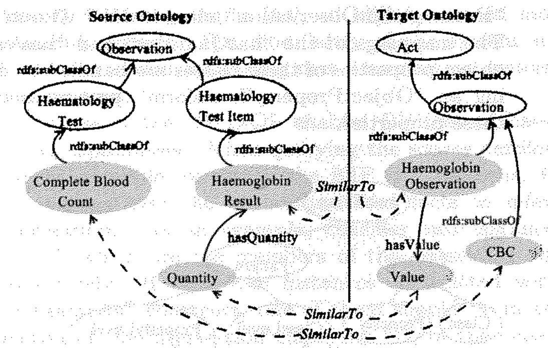


Figure 5: Mapping between HL7 v2 and HL7 v3 message structures

In the following sections, we describe how these tools realize the described functionality.

4. OWL MAPPING TOOL: OWLMT

We have developed an OWL mapping tool, called OWLmt, to handle ontology mediation by mapping the

OWL ontologies in different structures and with an overlapping content one into other. The architecture of the system, as shown in Figure 4, allows mapping patterns to be specified through a GUI tool based on a Mapping Schema. The Mapping Schema, as shown in Figure 7, is also defined in OWL.

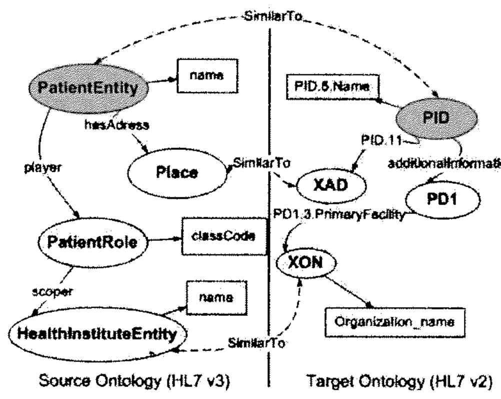


Figure 6: Mapping Object properties

Mapping patterns basically involve the following:

- *Matching the source ontology classes to target ontology classes*: In order to represent the matching between the classes of source and target ontologies, we have defined four mapping patterns: *EquivalentTo*, *SimilarTo*, *IntersectionOf* and *UnionOf*. Two identical classes are mapped through *EquivalentTo* pattern. *SimilarTo* implies that the involved classes have overlapping content. How the similar classes are further related is detailed through their data type properties and object properties by using "property mapping patterns". As an example, in Figure 5, the "HaemeglobinResult" class in HL7 v2 ontology is defined to be similar to "HaemoglobinObservation" class in HL7 v3 ontology. The mappings of the "hasQuantity" and "hasValue" object properties of these classes are handled by defining an "ObjectPropertyTransform" pattern between these properties.

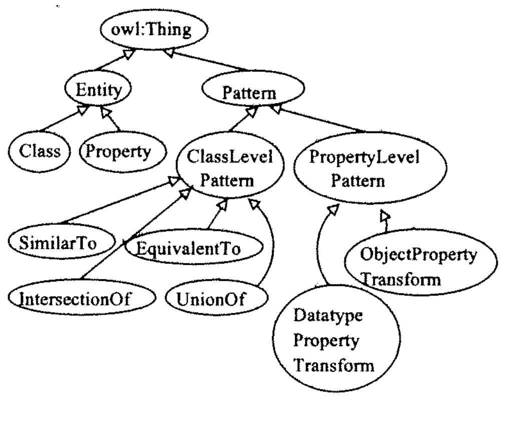


Figure 7: OWL Mapping Schema

The *IntersectionOf* pattern creates the corresponding instances of the target class as the intersection of the declared class instances. Similarly, the UnionOf pattern implies the union of the source classes' instances to create the corresponding instances of the target class. Furthermore, a class in a source ontology can be a more general (super class) of a class in the target ontology. In this case, which instances of the source ontology makes up the instances of the target ontology is defined through KIF (Knowledge Interchange Format) [15] conditions to be executed by the OWLmt mapping engine. When a source ontology class is a more specific (sub class) of a target ontology class, all the instances of the source ontology qualify as the instances of the target ontology.

- *Matching the source ontology Object Properties to target ontology Object Properties*: In addition to matching a single object property in the source ontology with a single object property in the target ontology, in some cases, more than one object properties in the source ontology can be matched with one or more object properties in the target ontology. Consider the example given in Figure 6. According to the HL7 v3 specifications, two entities, "Patient" and "HealthInstitueEntity" are connected by a "Role" class which is "PatientRole" in this case. On the other hand, in the target ontology, the "XON" class in HL7 v2.x represents the healthcare facility that a patient is registered. "PD1" (Patient Demographics 1) gives the patient information. "XON" is connected to the "PD1" by the "PD1.3.PrimaryFacility" object property. As it is clear from this example, relating a single object property in source ontology with a single object property in the target ontology does not suffice: There may be paths consisting of object property relations in the source and target ontologies that need to be mapped.

  OWLmt allows defining "ObjectPropertyTransform" pattern which represents the path of classes connected through object properties such that whenever a path defined in the source ontology (inputPath) is encountered in the source ontology instance, the path defined for target ontology (outputPath) is created in the target ontology instance. Paths are defined as triples in KIF [15] format and executed through the OWL-QL [21] engine. For example, assuming the path defined in the source ontology (Figure 6):

  (rdf:type ?x PatientEntity) (player ?x ?y)

  (rdf:type ?y PatientRole) (scoper ?y ?z)

  (rdf:type ?z HealthInstituteEntity).

  and assuming that it corresponds to the following path in the target ontology:

  (rdf:type ?x PID) (additionalInformation ?x ?y) (rdf:type ?y PD1) (PD1.3.PrimaryFacility ?y ?z) (rdf:type ?z XON)

  OWLmt constructs the specified paths among the instances of the target ontology in the execution step based on the paths defined among the instances of the source ontology.

- *Matching source ontology Data Properties to target ontology Data Properties*: Specifying the "Datatype-

PropertyTransform" helps to transform data type properties of an instance in the source ontology to the corresponding data type properties of instance in the target ontology. Since the data type properties may be structurally different in source and target ontologies, more complex transformation operations may be necessary than copying the data in source instance to the target instance. XPath specification [28] defines a set of basic operators and functions which are used by the OWLmt such as "concat", "split", "substring", "abs", and "floor". In some cases, there is a further need for a programmatic approach to specify complex functions. For example, the use of conditional branches (e.g. if-then-else, switch-case) or iterations (e.g while, for-next) may be necessary in specifying the transformation functions. Therefore, we have added JavaScript support to OWLmt. By specifying the JavaScript to be used in the "DatatypeProperty-Transform" pattern, the complex functions can also be applied to the data as well as the basic functions and the operators provided by XPath.

4.1 OWLmt Mapping Schema

The mapping patterns used in the OWLmt are defined through an OWL ontology called "Mapping Schema". Each mapping pattern is an owl:class in the "Mapping Schema" as shown in Figure 7. The additional information needed in the execution of the patterns are provided as KIF [15] expressions such as *inputPaths* and *outputPaths*. The *inputPath* and *outputPath* are data type properties of "ObjectProperty Transform Pattern" class and hold the query strings in the KIF format which are used in the execution to query the source ontology instances in order to build the target instances.

Each mapping relation specified through OWLmt GUI represents as an instance of these pattern classes, and the final the mapping definition is stored as an instance of the "Mapping Scheme" as a collection of pattern class instances.

In Figure 8, a part of the mapping definition of the example in Figure 6 is presented. First the *SimilarTo* relationship between the "Patient" and "PatientEntity" classes are represented with an instance of *SimilarTo* pattern. Then through an "ObjectPropertyTransform" pattern instance, the relationships between object properties linking the "PatientEntity" to "HealtInstituteEntity" classes and the object property linking the "PID" to "XON" classes are represented. Further details of the mapping tool are presented in [2].

This mapping definition is given as an input to the OWLmt Mapping Engine, which translates source ontology instances to target ontology instances.

4.2 OWLmt GUI

OWLmt GUI, as shown in Figure 9, consists of five components: Ontology Handler, Main Panel, Property Transformations Panel, Value Transformation Wizard and Object Property Definition Panel. The Ontology Handler is used in parsing and serializing the ontology documents. The class mapping patterns are defined in the main panel. The property mapping patterns are defined in the property transformation panel. This panel lets the user to create new property mapping patterns such as the "ObjectProperty-Transform" and "DatatypePropertyTransform". The value

```
<SimilarTo rdf:ID="SimilarTo_1">
 <similarToInput>
    <relatedTo rdf:resource=#PatientEntity/>
 </similarToInput>
 <similarToOutput>
    <relatedTo rdf:resource=#PID/>
 </similarToOutput>
 <operationName>PatientEntity_SimilarTo_PID</operationName>
</SimilarTo> .......

<ObjectPropertyTransform rdf:ID="ObjectPropertyTransform_1">
  <operationName>ObjectPropertyTransform_1</operationName>
  <includedIn rdf:resource=#SimilarTo_1/>
  <inputPath>(rdf:type ?x PatientEntity) (player ?x ?y)
       (rdf:type ?y PatientRole)   (scoper ?y ?z)
        (rdf:type ?z HealthInstituteEntity)
  </ inputPath>
  <outputPath>(rdf:type ?x PID) (additionalInformation ?x ?y)
          (rdf:type ?y PD1) (PD1.3.PrimaryFacility ?y ?z)
          (rdf:type ?z XON)
  </ outputPath>
</ObjectPropertyTransform>
```

Figure 8: An Example Mapping Definition

transformation wizard is used to configure a "DatatypePropertyTransform" pattern. By using this wizard, the functions used in the value transformation of the data type properties can be specified.

4.3 OWLmt Engine

The mapping engine is responsible for creating the target ontology instances using the mapping patterns given in the Mapping Definition and the instances of the source ontology. It uses OWL Query Language (OWL-QL) to retrieve required data from the source ontology instances. OWL-QL is a query language for OWL developed at the Stanford University [21]. While executing the class and property mapping patterns, the query strings defined through the mapping GUI are send to the OWL-QL engine with the URL of the source ontology instances. The query engine executes the query strings and returns the query results.

The OWL-QL engine uses the JTP (Java Theorem Prover) reasoning engine [14], an object-oriented modular reasoning system. The modularity of the system enables it to be extended by adding new reasoners or customizing existing ones.

The use of the OWL-QL enables OWLmt to have reasoning capabilities. When querying the source ontology instances or while executing the KIF [15] patterns, OWL-QL reasons over the explicitly stated facts to infer new information. As an example, consider two instances, I1 and I2, which are the members of the classes C1 and C2 respectively. If these two instances are related with the "owl:sameAs" construct, one of them should be in the extension of the intersection class, say C3, of the classes C1 and C2. Hence, the *IntersectionOf* pattern transforms the instance I1 and I2 to the instance I3 which is a member of C3 in the target ontology. However, assume that there is no direct "owl:sameAs" construct but there is a functional property which implies that these two instances are the same. The reasoning engine can infer from the definition of the "owl:FunctionalProperty" by using the rule:

```
(rdf:type ?prop owl:FunctionalProperty)
(?prop ?instance ?I1)
(?prop ?instance ?I2)
->
```

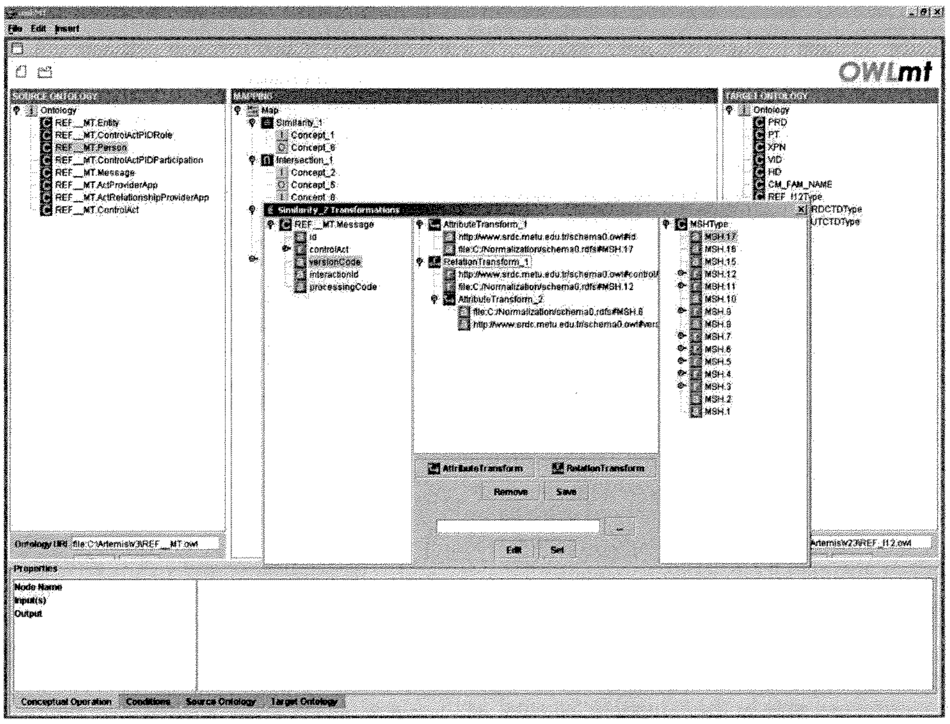


Figure 9: OWLmt GUI

(owl:sameAs ?I1 ?I2)

that the instances I1 and I2 are the same instance resulting in the instance I3 to be in the target ontology.

After executing the class mapping patterns, the mapping engine executes the property mapping patterns. Similar to the class mapping patterns, OWL-QL queries are used to locate the data. In order to perform value transformations, the mapping engine uses the JavaScripts in the "DatatypePropertyTransform" pattern. To execute the JavaScripts, an interpreter is used. The engine prepares the JavaScript by providing the values for the input parameters and sends it to the interpreter. The interpreter returns the result, which is then inserted as the value of the data type property in the target ontology instance.

5. EDI TO XML CONVERSION IN HL7

There are several commercial and open-source programming libraries that implement the HL7 standards. In our implementation, HAPI [6] (HL7 Application Programming Interface) Assembler/Disassembler Tool is used to transform the HL7 v2 EDI messages into their XML representations. HAPI provides open source libraries for parsing and manipulating both EDI and XML messages that are HL7 conformant. Furthermore the library enables message validation, that is, enforcement of HL7 data type rules for the values in the messages.

6. NORMALIZATION TOOL

As previously mentioned, currently the healthcare application messages are usually in XML or EDI format (which can be converted to XML). Hence there is a need for automatic bidirectional transformation of XML message instances to OWL message instances as well as automatic generation of OWL Schemas from XML Schema Definitions (XSDs). Such a transformation, called Normalization, has been realized within the scope of the Harmonise project [7].

The first step in the "Normalization" process is generating RDFS schemas from local XSD schemas. This step is called Conceptual Normalization (C-Normalization) phase where the C-Normalization engine parses the XML Schema, and using a set of predefined "Normalization Heuristics", creates the corresponding RDFS schema components for each XML Schema component automatically. Normalization Heuristics define how specific XML Schema construct can be projected onto a RDFS construct (entity or set of related entities) [7]. With this process, the complex type, element and attribute definitions of the XSD are represented as classes, and properties in the RDFS ontology. One of the "Normalization Heuristics" called "ComplexType2Class" projects each complex type definition in XSD onto a class definition in RDFS. Furthermore, the attribute definitions and the element definitions in XSD are converted to the "rdf:Property" by the "Attribute2Property" and "Element2Property" heuristics, respectively. After representing the complex types as classes and elements as properties, the domain and range of the properties are set. The "ElementParent2PropertyDomain" heuristic sets the domain of the property to the class which corresponds to the parent of the element in the XSD. Furthermore, the "ElementType2PropertyRange" heuristic sets the range of the property to the class which corresponds to the type of the element in the XSD as illustrated in Figure 10. The C-Normalization process produces a "Normalization Map" which defines the associations between the XML Schema and the re-engineered RDFS model. Further details of this work are available in [7].

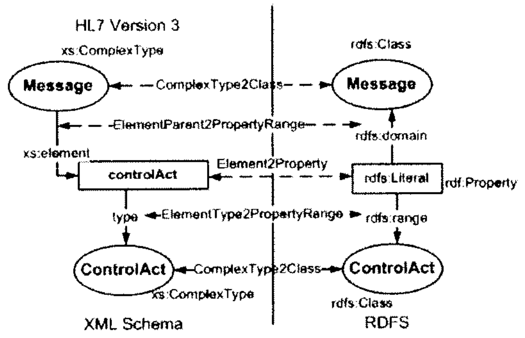


Figure 10: C-Normalization Phase

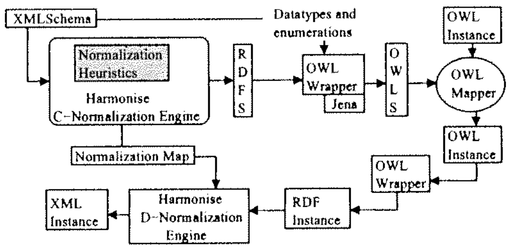


Figure 11: Normalization process for the bidirectional transformation of XML instances to OWL instances

The second step in "Normalization" is the Data Normalization Process (D-Normalization) which is used for transforming the data instances from XML to OWL or OWL to XML.

In Artemis architecture, we have used the Harmonise Normalization Engine. However since we need OWL Schemas instead of RDFS schemas, we developed an OWL wrapper using Jena API to create OWL schemas from the RDFS files after the C-Normalization step. Additionally in the D-Normalization step, through the same wrapper, the generated RDF instances are further translated in to OWL instances or vice versa as depicted in Figure 11.

Note that in Harmonise C-Normalization step, the enumeration of property values or basic data types defined in XML Schemas cannot be preserved. To handle this, the OWL Wrapper developed carries the enumeration of property values and basic data types to the OWL Schema. The enumerated classes are represented using <owl:oneOf rdf:parseType="Collection"> construct in case of enumerated classes, and using <owl:oneOf> and <rdf:List> constructs in case of enumerated data types. The data types are represented by referring to XML Schema data types using RDF data typing scheme.

7. RELATED WORK

Providing the interoperability of heterogeneous information systems through ontology mediation has been an active research area recently.

In [17], an RDF (Resource Description Framework) mapping meta-ontology, called RDF Translation (RDFT), is proposed which specifies a language for mapping XML DTDs to and from RDF Schemas for business integration tasks.

The Harmonise project [7] developed a harmonization network for the tourism industry to allow participating tourism organisations to keep the proprietary data format and use ontology mediation while exchanging information in a seamless manner. For this purpose they have defined a Interoperability Minimum Harmonization Ontology (IMHO) and an interchange format for tourism industry. The MAFRA tool is used for ontology mediation [16]. MAFRA uses a component that defines the relations and transformations between RDF ontologies. For representing the similarities in a formal way, MAFRA provides a meta-ontology called Semantic Bridge Ontology (SBO).

Although OWLmt has a similar approach for representing mapping definitions, OWLmt maps OWL ontologies instead of RDF ontologies and OWLmt engine uses OWL-QL to execute the KIF expressions defined in mapping patterns to retrieve the ontology instances and hence has a reasoning capability.

In [22], an extensible language, called an XML-based extensible Ontology Mapping Language (XeOML) is proposed for describing mappings between domain ontologies. XeOML is defined by an XML schema, called AbstractMapping, which provides information for describing mappings between ontologies, detailing the structure of a mapping document and defining the set of elements that populate an ontology.

In OWLmt, the mapping definition itself is also an OWL instance rather than XML. Additionally, OWLmt provides a graphical interface through which this ontology mapping can be created and a mapping engine which interprets the mapping definition in order to automatically create target ontology instances.

[25] also focuses on integration of heterogeneous data sources in the Semantic Web context using a semantic mediation approach based on ontologies. They use OWL to formalize ontologies of different resources and to describe their relations and correspondences to allow the semantic interoperability between them. They propose an architecture based on mediator-wrapper component. The relationships between local ontologies is defined in OWL, i.e, OWL is used as a mapping definition language exploiting native OWL constructs such as equivalantClass and equivalentProperty. The mediator queries the local ontologies wrapping the backend information systems by using the mapping definition defined in OWL to mediate between heterogeneous local data representations. This approach is limited to the mapping definition capabilities of native OWL constructs.

In OWLmt we have defined a mapping schema definition in OWL including several mapping patterns, which are interpreted by the OWLmt engine for handling the mapping process.

8. CONCLUSIONS

One of the most challenging problems in the healthcare domain today is providing interoperability among healthcare information systems. In order to tackle this problem, we propose an engineering approach to semantic interoperability within the scope of the Artemis project. For this purpose, the existing applications are wrapped as Web services and the messages they exchange are annotated with OWL ontologies which are then mediated through an ontology mapping tool developed, namely, OWLmt. One of the major contributions of the OWLmt is the use of OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the graphically defined mapping patterns.

Although the platform proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use in the healthcare domain, we have chosen to mediate between HL7 version 2 and HL7 version 3 messages to demonstrate the functionalities of the proposed platform. Since neither version 2, nor version 3 messages are ontology instances, these message structures, EDI and XML respectively, are normalized to OWL before OWL mapping process. Additional tools exploited for this purpose are also discussed in the paper.

9. REFERENCES

[1] Artemis A Semantic Web Servicebased P2P Infrastructure for the Interoperability of Medical Information Systems, http://www.srdc.metu.edu.tr/-webpage/projects/artemis/.

[2] Bicer, V., "OWLmt: OWL Mapping Tool", M.Sc. Thesis, Dept. of Computer Eng., METU, in preparation.

[3] Castor's XMLInstance2Schema tool, http://castor.exolab.org/.

[4] CEN/ISSS eHealth Standardization Focus Group Report, "Current and future standardization issues in the eHealth domain: Achieving interoperability", Part One: Main text, Draft V4.1, 20040816.

[5] ENV 13606:2000 "Electronic Healthcare Record Communication", http://www.centc251.org/TCMeet/-doclist/TCdoc00/N00048.pdf.

[6] HL7 Application Programming Interface (HAPI), http://hl7api.sourceforge.net

[7] Harmonise, IST200029329, Tourism Harmonisation Network, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.

[8] Health Level 7, http://www.hl7.org.

[9] HL7, Chapter 11 Patient Referral, http://www.hl7.org/library/General/v231.zip

[10] HL7 Reference Information Model (RIM), http://www.hl7.org/library/data-model/RIM/-modelpage_mem.htm.

[11] HL7 v3 Schema Generator, http://www.hl7.org/-library/data-model/V3Tooling/toolsIndex.htm

[12] ISO/TS Health Informatics - Requirements for an electronic health record architecture, Technical Specification, International Organization for Standardization (ISO), Geneva, Switzerland, 2004.

[13] Jena Framework, http://jena.sourceforge.net/ .

[14] Java Theorem Prover (JTP), http://www.ksl.stanford.edu/software/JTP/ .

[15] Knowledge Interchange Format (KIF), http://logic.stanford.edu/kif/kif.html .

[16] A. Maedche, D. Motik, N. Silva, R. Volz, "MAFRA-A MApping FRAmework for Distributed Ontologies", In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management EKAW-2002, Madrid, Spain, 2002.

[17] B. Omelayenko, D. Fensel, C. Bussler, "Mapping Technology for Enterprise Integration", Proc. of Special Track on Semantic Web at The 15th International FLAIRS Conference, USA, May 2002.

[18] OpenEHR Foundation, http://www.openehr.org/ .

[19] Web Ontology Language (OWL), http://www.w3.org/TR/owlfeatures/.

[20] OWL Mapping Tool (OWLmt), http://www.srdc.metu.edu.tr/ artemis/owlmt/

[21] OWL Query Language, http://ksl.stanford.edu/projects/owlql/

[22] M. T. Pazienza, A. Stellato, M. Vindigni, F. M. Zanzotto, "XeOML: An XML-based extensible Ontology Mapping Language", Workshop on Meaning Coordination and Negotiation, held in conjunction with 3rd International Semantic Web Conference (ISWC-2004) Hiroshima, Japan, November 8, 2004

[23] Resource Description Framework Schema (RDFS), http://www.w3.org/TR/rdfschema/.

[24] Rose Tree, http://www.hl7.org/library/data-model/-V3Tooling/toolsIndex.htm

[25] S. Suwanmannee, D. Benslimane and Ph. Thiran, "OWL-based Approach for Semantic Interoperability", in the Proceedings of AINA, IEEE Computer Science Press, March 2005.

[26] Thompson TG, Brailer DJ. "The decade of health information technology: delivering consumer-centric and information-rich health care: framework for strategic action". Washington (DC): Department of Health and Human Services, National Coordinator for Health Information Technology; 2004, http://www.hhs.gov/-onchit/framework/hitframework.pdf.

[27] XML encoding rules of HL7 v2 messages - v2.xml, http://www.hl7.org/Special/Committees/xml/drafts/-v2xml.html

[28] XML Path Language, http://www.w3.org/TR/xpath

Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics [a]

Asuman Dogac*[1], Gokce B. Laleci[1], Yildiray Kabak[1], Seda Unal[1], Thomas Beale[2], Sam Heard[2], Peter Elkin[3], Farrukh Najmi[4], Carl Mattocks[5], David Webber[6]

Middle East Technical University, Turkey [1]
Ocean Informatics, Australia [2]
Mayo Clinic, USA [3]
Sun Micro Systems, USA[4]
OASIS ebXML Registry SCM SC, USA[5]
OASIS CAM TC, USA[6]
*Corresponding author

Abstract: Using archetypes is a promising approach in providing semantic interoperability among healthcare systems. To realize archetype based interoperability, the healthcare systems need to discover the existing archetypes based on their semantics; annotate their archetypes with ontologies; compose templates from archetypes and retrieve corresponding data from the underlying medical information systems.

In this paper, we describe how ebXML Registry semantic constructs can be used for annotating, storing, discovering and retrieving archetypes. For semantic annotation of archetypes, we present an example archetype metadata ontology and describe the techniques to access archetype semantics through ebXML query facilities. We present a GUI query facility and describe how the stored procedures we introduce, move the semantic support beyond what is currently available in ebXML registries.

We also address how archetype data can be retrieved from clinical information systems by using ebXML Web services. A comparison of Web service technology with ebXML messaging system is provided to justify the reasons for using Web services.

Keywords: Healthcare Informatics; Semantic Interoperability; ebXML Registries; Archetypes.

Reference to this paper should be made as follows: Dogac, A., Laleci, G.B., Kabak, Y., Unal, S., Beale, T., Heard, S., Elkin, P., Najmi, F., Mattocks, C., Weber, D. (xxxx) 'Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics', *Intl. Journal of Metadata, Semantics and Ontologies*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Asuman Dogac is a full professor at the Computer Engineering Dept., Middle East Technical University (METU), Ankara, Turkey and the founding director of the Software Research and Development Center (SRDC), METU. Her research interests include healthcare informatics, Web services and semantic interoperability.

Gokce B. Laleci is a PhD candidate at METU, Dept. of Computer Eng. and a senior researcher at SRDC. Her research interests include semantic Web, Web service technology and healthcare informatics.

Yildiray Kabak is a PhD candidate at METU, Dept. of Computer Eng. and a senior researcher at SRDC. His research interests include Web service registries, peer-to-peer computing and healthcare informatics.

Seda Unal is a MSc student at the Computer Engineering Dept., METU, and is a researcher at SRDC. Her research interests include Web service composition technologies and healthcare informatics.

Thomas Beale (BSc, BEng, EE) has participated in international standards work (OMG HDTF, HL7, CEN TC 251). He has been instrumental in the creation and evolution of the openEHR Foundation, now an internationally recognised effort informing various national EHR projects and standards work.

Sam Heard is practicing clinician and an adjunct Professor in Health Informatics at Central Queensland University in Australia and the director of the openEHR Foundation.

Peter Elkin is a Professor of Medicine, Director of Laboratory of Biomedical Informatics at the Department of Internal Medicine, Mayo Medical School, USA. He is the co-chair of HL7 Template Special Interest Group.

Farrukh Najmi is an XML Standards Architect at Sun Microsystems, Inc. USA. His focus in recent years has been on Registries, Repositories, Enterprise Content Management, and eGovernment. He is an active contributor to the ebXML standard and is a principal author and editor of the OASIS ebXML Registry specifications.

Carl Mattocks is the founder and CEO of CHECKMi, a New Jersey-based company developing agent-based software. He is a long time contributor to the development and exploitation of metadata-centric specifications. Currently he acts as a co-chair for the Business Centric Methodology TC and the ebXML Registry Semantic Content SC.

David RR Webber holds two US Patents on advanced information transformation with EDI. Currently he is chairing the OASIS Technical Committee work developing XML Content Assembly Mechanism (CAM) specifications including facilitating the development and deployment of semantic registry systems by government and industry organizations.

---

* This work is supported by the European Commission through IST-1-002103-STP Artemis project and in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 104E013

1 Introduction

Most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. To make the matters worse, a patient's health information may be spread out over a number of different institutes which do not interoperate. This makes it very difficult for clinicians to capture a complete clinical history of a patient.

A number of standardization efforts are progressing to provide the interoperability of healthcare systems such as CEN TC 251 prEN13606 [10], openEHR [37] and HL7 Version 3 [28]. However, exchanging machine processable electronic healthcare records have not yet been achieved. For example, although HL7 Version 2 Messaging Standard is the most widely implemented standard for healthcare information in the world today, being HL7 Version 2 compliant does not imply direct interoperability between healthcare systems. This stems from the fact that Version 2 messages contain many optional data fields. This optionality provides great flexibility, but necessitates detailed bilateral agreements among the healthcare systems to achieve interoperability. To remedy this problem, HL7 [25] has developed Version 3 which is based on an object-oriented data model, called Reference Information Model (RIM) [27].

Yet, given the large number of standards in the healthcare informatics domain, conforming to a single standard does not solve the interoperability problem.

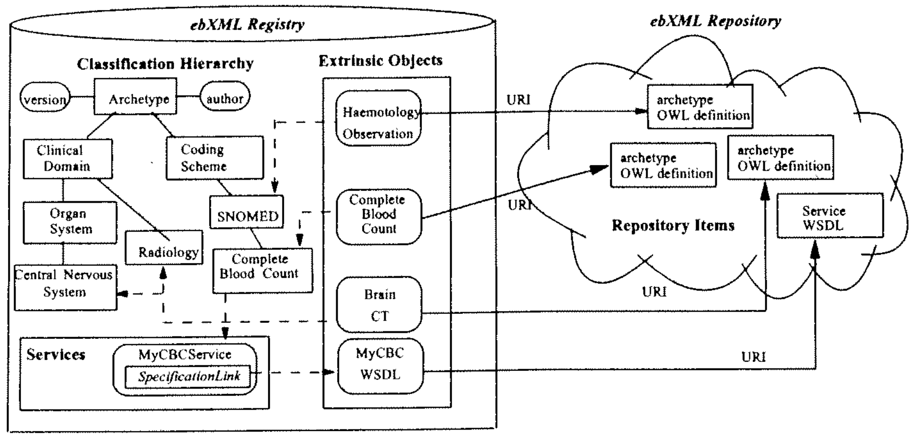


Figure 1 Handling Archetype Semantics in ebXML Registries

In fact, the full shareability of data and information requires two levels of interoperability:

- *The Functional (syntactic) interoperability* which is the ability of two or more systems to exchange information. This involves agreeing on the common network protocols such as Internet or Value Added Networks; the transport binding such as HTTP, FTP or SMTP and the message format like ASCII text, XML (Extensible Markup Language) or EDI (Electronic Data Interchange). One of the successful examples of functional interoperability is electronic mail: an email can be sent and received by any platform due to the well established standards for the transport binding which is SMTP running on TCP/IP and the message format which is ASCII or HTML. That is, SMTP and the fixed message format makes it possible for two systems exchange emails. However, note that the message content is only for human consumption.

- *Semantic interoperability* is the ability for information shared by systems to be understood at the level of formally defined domain concepts so that the information is computer processable by the receiving system [31]. In other words, semantic interoperability requires the semantics of data to be defined through formally defined domain specific concepts. Semantic interoperability in the healthcare domain can be achieved by conforming to a single healthcare information standard, such as HL7 Version 3. However, it is not realistic to expect all the healthcare institutes to conform to a single standard. A promising approach in providing interoperability among different standards in the healthcare domain is the archetypes [6]. An "archetype" is a syntactically and semantically structured aggregation of vocabulary or other data that is the basic unit of clinical information [20]. When healthcare systems start exchanging information with well defined syntax and semantics as proposed by "archetypes", semantic interoperability among them will become a reality.

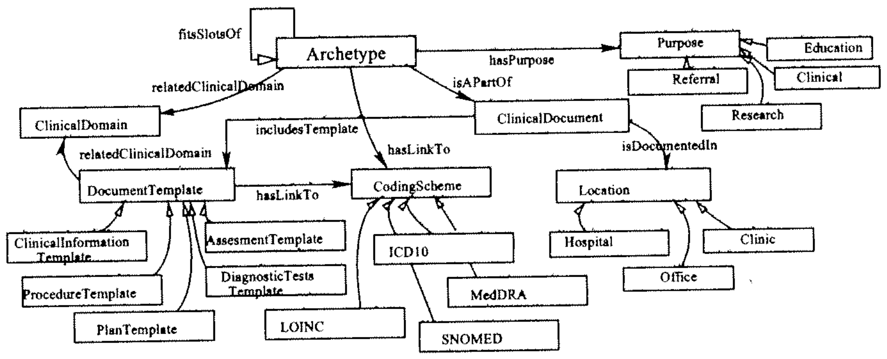


Figure 2 An Archetype Metadata Ontology

An important aspect of archetype based interoperability is providing the ability to the healthcare institutes and systems to share archetypes and their metadata among them. In this paper, we describe how ebXML registries can be used for handling templates and archetypes by using the semantic constructs of the ebXML registry. This work is carried out within the scope of Artemis project [3] which aims to provide interoperability in the healthcare domain through semantically enriched Web services.

Electronic Business XML (ebXML) [14] is a standard from OASIS[36] and United Nations Centre for Trade Facilitation and Electronic Business, UN/CEFACT [43]. ebXML specifies an infrastructure that allows enterprises to find each other's services, products, business processes and documents in a standard way and thus helps to facilitate conducting electronic business. One of the important characteristics of ebXML compliant registries is that they provide standard mechanisms both to define and to associate metadata with registry entries.

In order to achieve archetype based interoperability among healthcare institutes through ebXML registries, we propose the following phases:

- *Semantically annotating the archetypes*: For healthcare systems to exchange archetypes, they need to discover the archetypes of the institutes they wish to communicate with. This discovery must be based on the semantics of archetypes such as the purpose of the archetype, the clinical domains it is associated with, the types of clinical documents it is used in as constituents, where they fit into the slots of other archetypes as well as authorship, and the version of the archetype. Since ebXML registry allows metadata to be associated with the registry entries and provides mechanisms for semantic based discovery of registry entries, it provides a convenient medium for handling archetypes.

  As the first phase of archetype based interoperability, we show how ebXML registry semantic constructs can be used to annotate archetypes. For this purpose, we present an example archetype metadata ontology. It should be noted that our purpose is not to propose an archetype ontology but rather to show how it can be exploited once it is specified by standard bodies.

- *Retrieving archetypes from the registry through ebXML query facilities*: The archetypes need to be discovered in the registry according to their semantics. This semantic information serves the purpose of how or where the archetype can be used. For example, the metadata can be queried to find out the archetypes associated with a given clinical domain, or the coding schemes they are referring to, or the authorship.

  We show how the semantic information used in annotating the archetypes in an ebXML registry can be queried through ebXML standard query mechanisms such as Filter Query or SQL-92 query. In order to facilitate access to the system by novice users, we present ready to be used stored queries for discovering archetypes according to their metadata.

  We then describe how Web Ontology Language (OWL) can be used in ebXML registries to enhance the archetype semantics. Although the ebXML semantic mechanisms are useful as they stand, currently, semantics is becoming a much broader issue than it used to be through the use of ontologies and standard ontology definition languages [11, 21, 35]. An important standard ontology language is W3C's Web Ontology Language (OWL) [44]. There are several opportunities to be gained in extending ebXML semantic mechanisms to standard ontology languages, such as OWL. In this way, it becomes possible to exploit the richer semantic constructs of OWL as well as its reasoning capabilities.

  In our previous work, we describe how ebXML registries can be enriched with OWL semantics [12] and how this additional semantics can be processed with stored queries [13]. In this paper, we show how archetype and template semantics can benefit from OWL aware ebXML registries.

- *Retrieving archetype data from medical information systems*: After discovering archetypes by exploiting their metadata, it is also necessary to retrieve the associated information from the medical information systems conforming to the templates and archetypes.

  Although the templates and archetypes semantically and structurally define the data to be retrieved through formally defined domain concepts and hence provide for the semantic interoperability, we still need to provide the functional level interoperability. As mentioned previously functional interoperability involves agreeing on the common network protocols, the transport bindings and the message formats. Although ebXML messaging provides for this, we choose to use another functional interoperability standard, namely, Web services which is also supported by ebXML registries.

The paper is organized as follows: In Section 2, the objectives of archetypes and templates are summarized and how semantic interoperability can be achieved through archetypes is described. In Section 3, we show how archetype semantics can be handled in ebXML registries. Here we also describe enhancing ebXML registries with Web Ontology Language (OWL) semantics and explain how archetypes semantics can benefit from OWL aware ebXML registries. In Section 4, we present an example archetype metadata ontology to annotate archetypes and templates. Section 5 describes the techniques to access archetype semantics through ebXML query facilities. In this section we introduce a GUI query facility and show how stored procedures move the semantic support beyond what is currently available in ebXML registries. Section 6 addresses retrieving data from clinical information systems conforming to archetypes. In this section, we compare Web service technology with ebXML messaging system and provide the reasons for choosing Web services in accessing the underlying medical information systems. In Section 7, we describe how archetypes can be composed through Web service composition techniques. Finally, Section 8 concludes the paper.

2 Why do we need Templates and Archetypes?

Healthcare is one of the few domains where sharing information is the norm, rather than the exception [22]. Archetypes and templates aim to solve the semantic interoperability problem in the healthcare domain by allowing modeling of domain concepts based on reference models, independent from the information systems. Archetypes have been adopted by a number of standards such as openEHR [37], CEN TC/251 [10] and is considered to be used by HL7 [25] as a basis for its templates specification.

An archetype is a reusable, formal expression of a distinct, domain-level concept such as "blood pressure", "physical examination", or "laboratory results", expressed in the form of constraints on data whose instances conform to some reference model [6]. The reference model refers to any model such as CEN TC 251 prEN13606 [10], openEHR [37], or the HL7 CDA schema [26]. The key feature of the archetype approach to computing is a complete separation of information models (such as object models of software, models of database schemas) from domain models [6].

A formal language for expressing archetypes has been introduced which is called Archetype Definition Language (ADL) [2]. In Figure 3, a part of "Complete Blood Count" archetype definition is presented in ADL. The complete ADL definition can be found in [9]. Here the "OBSERVATION" class from the reference information model is restricted to create "Complete Blood Count" archetype, by restricting its CODED_TEXT value to "ac0001" term, (ac0001 term is defined to be "complete blood count" in the constraint_definitions part of the ADL, and declared to be

```
OBSERVATION[at1000.1] matches {-- complete blood picture
 name matches {
     CODED_TEXT matches {
         code matches {[ac0001]}  -- complete blood count}}
 data matches {
     LIST_S[at1001] matches {-- battery
        items cardinality matches {0..*} \epsilon {
          ELEMENT[at1002.1] matches {-- haemaglobin
               name matches {
                 CODED_TEXT matches {
                     code matches {[ac0003]} -- haemaglobin}}
               value matches {
                  QUANTITY matches {
                     value matches {0..1000}
                     units matches {^g/l|g/dl|.+^}}}}
          ELEMENT[at1002.2] occurrences matches {0..1} matches
      {-- haematocrit
              name matches {
                 CODED_TEXT matches {
                     code matches {[ac0004]}-- haematocrit}}
              value matches {
                  QUANTITY matches {
                      value matches {0..100}
                      units matches {"%"}}}}
          ELEMENT[at1002.3] occurrences matches {0..1} matches
      {-- platelet count
              name matches {
                 CODED_TEXT matches {
                     code matches {[ac0005]} -- platelet count}}
              value matches {
                  QUANTITY matches {
                      value matches {0..100000}
                      units matches {"/cm^3"}
                     }}}}}}}
```

Figure 3 The ADL definition of "Complete Blood Count" Archetype

equivalent to Loinc::700-0 term in the term bindings part), and by defining its content to be a list of "Haemoglobin", "Haematocrit" and "Platelet Count" test result elements.

A template on the other hand is a directly, locally usable data creation/validation artefact which is semantically a constraint/choice on archetypes, and which often corresponds to a whole form or a screen. Templates in general have a 1:N relationship with underlying concepts, each of which is described by an archetype.

Note that in order to address the interoperability problem through archetypes, a generic approach based on a "harmonised" information model needs to be adopted [7]. Alternatively, the reference models of these standardization bodies (openEHR, CEN TC/251, HL7) can be represented through an ontology language like OWL. Then by defining the archetypes constraining these reference models also in OWL, and by providing the mapping between these reference models through ontology mapping, the interoperability of the archetype instances can be achieved automatically.

3 What Does ebXML Registry Provide for Archetypes?

For healthcare systems to exchange information in an interoperable manner, they need to discover the archetypes and templates and their associated semantics. ebXML Registry, through its semantic constructs, provides an efficient medium to annotate, store, discover and reuse of archetypes.

In the following sections, we first briefly present ebXML Registry architecture and then demostrate how basic semantic constructs of ebXML Registry can be used for this purpose. Then, we present how ebXML registries can be enhanced with OWL semantics and how this knowledge can be exploited for handling arhetypes.

3.1 ebXML Specification

ebXML facilitates electronic business as follows:

- In order for enterprises to conduct electronic business with each other, they must first discover each other and the products and services they have to offer. ebXML provides a registry where such information can be published and discovered.
- An enterprise needs to determine which business processes and documents are necessary to communicate with a potential partner. *A Business Process Specification Schema (BPSS)* in ebXML, provides the definition of an XML document that describes how an organization conducts its business.
- After this phase, the enterprises need to determine how to exchange information. The Collaboration Protocol Agreement (CPA) specifies the details of how two organizations have agreed to conduct electronic business.

A registry can be established by an industry group or standards organization. A repository is a location (or a set of distributed locations) where a document pointed at by the registry reside and can be retrieved by conventional means (e.g., http or ftp).

It should be noted that within the scope of this paper, we address how to handle the archetypes and templates and not the healthcare business processes. There are other healthcare informatics initiatives complementary to our work such as IHE IT Infrastructure Integration Profiles [23] which define some of the business processes in the healthcare domain and IHE Cross-Enterprise Clinical Documents Sharing (XDS) [24] which specify how to use ebXML registries for healthcare document sharing. Note however that semantic issues are not yet addressed by these initiatives.

3.2 ebXML Registry Architecture and Information Model

ebXML registry provides a persistent store for registry content. The current registry implementations store registry data in relational databases. ebXML Registry Services Specification defines a set of Registry Service interfaces which provide access to registry content. There are a set of methods that must be supported by each interface. A registry client program utilizes the services of the registry by invoking methods on one of these interfaces. The Query Manager component also uses these methods to construct the objects by obtaining the required data from the relational database through SQL queries. In other words, when a client submits a request to the registry, registry objects are constructed by retrieving the related information from the database through SQL queries and are served to the user through the methods of these objects [19].

An ebXML registry [18] allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "classification" mechanism. This information can then be used to discover the registry objects by exploiting the ebXML query mechanisms.

Table 1 Predefined Association Types in ebXML Registries

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.

3.3 Exploiting ebXML Registries for Semantically Annotating Archetypes

As previously noted, ebXML Registry semantic constructs can be used to annotate registry objects. For example, as shown in Figure 1, the archetype "HaemotologyObservation" can be stored as an Extrinsic Object (which is also a Registry Object) whose link points to the repository item that holds the actual archetype definition. The metadata of the archetype, on the other hand, can be classified with as many *ClassificationNodes* as needed to describe its semantics. For example, in Figure 1, "HaemotologyObservation Archetype" Extrinsic Object is classified with *SNOMED* [39] clinical coding terms. Therefore when a user wishes to find all the archetypes in the registry annotated with SNOMED Clinical Coding terms; issuing an ebXML query will suffice. In other words, by relating a *ClassificationNode* (such as *SNOMED*) with a RegistryObject in ebXML (such as "HaemotologyObservation"), we make this object an implicit member of the *SNOMED* node and hence the object inherits the semantics associated with that node. The ebXML query to find all the archetypes coded with *SNOMED*, first finds the *SNOMED ClassificationNode* and the links from this node. These links are then used to retrieve the related ExtrinsicObjects and by issuing a "getContentQuery", the content of the archetypes are retrieved from the Repository. In fact such queries can be automatically generated and issued through Graphical User Interfaces as described in Section 5.

Through this example, we describe the simplest and most basic way of associating semantics with ebXML registry objects. ebXML classification hierarchies allow more complex semantics to be defined and queried both through the "slot" mechanism and through the predefined associations between registry objects. For example, through "slot" mechanism, it is possible to define the properties of classes (ClassificationNodes). "Slot" instances provide a dynamic way to add arbitrary attributes to "RegistryObject" instances. For the example shown in Figure 1, the archetype properties such as "version" and "authorship" are defined by using slots.

Furthermore, through predefined associations, it is possible to associate ClassificationNodes. There are a number of predefined *Association Types* that a registry must support to be ebXML compliant [18] as shown in Table 1.

Although ebXML semantic mechanisms are useful as they stand, currently, semantics is becoming a much broader issue than it used to be and the trend is to use ontologies [11, 21, 35]. One of the driving forces for ontologies is the Semantic Web initiative [5]. As a part of this initiative, W3C's Web Ontology Working Group defined Web Ontology Language (OWL) [44].

There are several opportunities to be gained in extending ebXML semantics mechanisms to make them OWL aware. In this way, it will become possible to represent ontologies defined in OWL in the ebXML registry and to exploit the richer semantic constructs of OWL as well as its reasoning capabilities.

In our previous work, we describe how ebXML registries can be enriched with OWL semantics [12] and how this additional semantics can be processed with stored queries [13]. Here, for the sake of completeness, we provide a brief summary.

Being OWL aware entails the following enhancements to the ebXML registry:

- *Representing OWL constructs through ebXML constructs*: ebXML provides a classification hierarchy made up of classification nodes and predefined type of associations between the registry objects. We represent *OWL* constructs by using combinations of these constructs and define additional types of associations when necessary. For example, 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 the target object of "rdfs:domain" property can be mapped to a "source RegistryObject" and the target object of "rdfs:range" can be mapped to a "target RegistryObject". "OWL ObjectProperty","DataTypeProperty" and "TransitiveProperty" are defined by introducing new association types such as "objectProperty". ebXML specification allows additional associations to be defined.

  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.

As another example, in ebXML, the predefined "EquivalentTo" association (Table 1) expresses the fact that the source registry object is equivalent to target registry object. Therefore, "EquivalentTo" association is used to express "owl:equivalentClass", "owl:equivalentProperty" and "owl:sameAs" properties since classes, properties and instances are all ebXML registry objects. The details of how the rest of the OWL constructs are represented in ebXML registries is available from [13].

- *Automatically generating ebXML constructs from the OWL descriptions and storing the resulting constructs into the ebXML registry*: We developed a tool to create an ebXML Classification Hierarchy from a given OWL ontology automatically by using the transformations described. The OWL file is parsed using Jena [33], 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. For example the OWL definition of Archetype Metadadata Ontology presented in Figure 2 is parsed and a Classification Hierarchy, a part of which is presented in Figure 1, is created automatically in the ebXML registry.

- *Querying the registry for enhanced semantics*: When various constructs of OWL are represented by ebXML classification hierarchies, although some of the OWL semantics stored in an ebXML registry can be retrieved from the registry through ebXML query facilities, further processing needs to be done by the application program to make use of the enhanced semantics.

  For example, two classification nodes can be declared as equivalent classes through the "EquivalentTo" predefined "association" in the ebXML registry. To make any use of this semantics, given a query requesting instances of a class, the application program must have the necessary code to find out and retrieve also the instances of classes which are declared to be equivalent to this class.

  To relieve the users from this burden, the code to process the OWL semantics can be stored in ebXML registry architecture through predefined procedures. As an example, the stored procedure given in Figure 4 retrieves all the archetype instances of a ClassificationNode (class) as well as the archetype instances of all classes equivalent to this class.

As an example to how such a stored query might help the users, assume that SNOMED "CompleteBloodCount" term is defined to be equivalent to the 'Full Blood Count" term in another coded term list, say, MedDRA (Medical Dictionary for Regulatory Activities) [34]. Then through the stored procedure given in Figure 4, when a user wishes to retrieve the archetype instances related with "CompleteBloodCount" term, it becomes possible to automatically obtain the archetype instances that are classified with SNOMED as well as those instances classified with MEdDRA "Full Blood Count" term.

```
CREATE PROCEDURE findEquivalentInstances($className)
BEGIN
SELECT N.value FROM ExtrinsicObject EO, Name_ N
    WHERE EO.id IN (
    SELECT classifiedObject
    FROM Classification
    WHERE classificationNode IN (
      SELECT id
      FROM ClassificationNode
      WHERE id IN (
          SELECT parent
          FROM name_
          WHERE value LIKE $className
          )
      UNION
      SELECT A.targetObject
      FROM Association A, Name_ N, ClassificationNode C
      WHERE A.associationType LIKE 'EquivalentTo' AND
            C.id = N.parent AND
            N.value LIKE $className AND
            A.sourceObject = C.id
          )
    ) AND EO.id=N.parent
END;
```

Figure 4 A Stored Procedure to Find Equivalent Instances

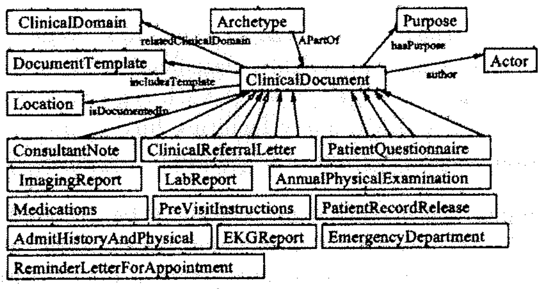


Figure 5 The Clinical Document Ontology

4 Defining Archetype and Template Metadata

An archetype is a reusable, formal expression of a distinct, domain-level concept such as "blood pressure", "physical examination", "laborotory results", expressed in the form of constraints on data whose instances conform to some class model, known as a reference model [6]. An OpenEHR template on the other hand is a directly, locally usable data creation/validation artefact which is semantically a constraint/choice on archetypes, and which will often correspond to a whole form or screen [6]. In this section, we describe an example Archetype Metadata Ontology to annotate archetypes and templates. It should be noted that our purpose is not to propose an archetype ontology but rather to show how it can be exploited once it is specified by standard bodies.

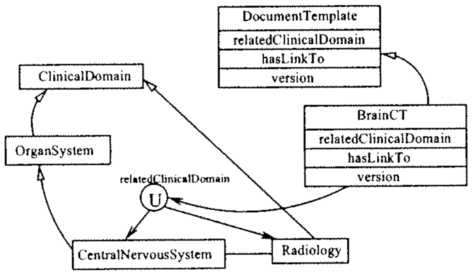


Figure 6 The BrainCT Template

4.1 An Archetype Ontology for Semantically Annotating Archetypes

We propose to classify archetypes on the basis of the following semantic information:

- The coding schemes it is referring to
- The purpose of the archetype
- The clinical domains it is associated with
- The types of clinical documents it is used in as constituents
- Other archetypes into whose slots it fits

The first level of the ontology covers this semantics as properties of the "Archetype" class as presented in Figure 2. For example, "Archetype" class has a property named "hasLinkTo" whose range is the class "CodingScheme". The "CodingScheme", "Purpose", "ClinicalDocument", "Clinical Domain" and "DocumentTemplate" classes are further detailed by defining their subclasses and properties.

The Clinical Document class in Figure 2 is organized on the basis of following axes as shown in Figure 5: author, location where the document is reported, purpose of the document, constituent templates, and clinical domains the document is related with. These metadata are represented as the properties of the "ClinicalDocument" class. A number of Clinical Document types are created as subclasses of this class.

Similarly "DocumentTemplate" class in Figure 2 is associated with "CodingScheme" class to indicate the coding schemes it is referring to, and it is associated with the "ClinicalDomain" class to indicate with which clinical domains the template is related.

As an example, "BrainCT" template can be created as a subclass of "DocumentTemplate" class, and the range of the "relatedClinicalDomain" property can be restricted to the "CentralNervousSystem" and "Radiology" classes (which are created as subclasses of "ClinicalDomain" class) as presented in Figure 6. In this way it is possible to query this template by referring to the "CentralNervousSystem" and "Radiology" domains.

Additionally generic template types "AssessmentTemplate", "ClinicalInformationTemplate", "PlanTemplate", "ProcedureTemplate" and "DiagnosticTestsTemplate" are added to the metadata ontology as subclasses of "DocumentTemplate"

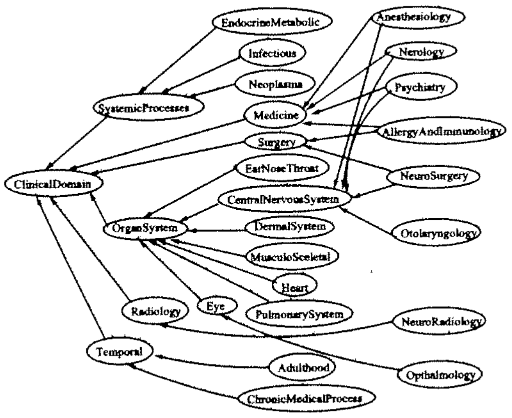


Figure 7 The Clinical Domain Ontology

class as shown in Figure 2. Finally the Clinical Domain hierarchy is detailed in this metadata Ontology as depicted in Figure 7. "ClinicalDomain" class is defined to have subclasses such as "OrganSystems", "SystemicProcesses", "Temporal", "Medicine","Surgery" and "Radiology". Leaf level clinical domains are defined as subclasses of one or more of these classes. As an example, "Neurosurgery" domain is defined as a subclass of both "CentralNervousSystem" (which is a subclass of "OrganSystems" class), and "Medicine" classes.

5 How to access archetype metadata through ebXML query facilities?

In this section we describe how archetypes can be represented and accessed in ebXML registries. Archetypes and their semantics are represented in an ebXML registry as follows:

- The "archetype metadata ontology" is stored in the ebXML registry to be used for querying the archetype definitions. For this purpose, a "SubmitObjectsQuery" is created by parsing the ontology and sent to the registry, which in turn creates a classification hierarchy as presented in Figure 8 (a). As described in Section 3.3, OWL classes are represented as "Registry Information Model (RIM) Classification Nodes" and OWL properties are represented as "RIM Associations".
- An "archetype" is represented in the Registry as a "RIM Extrinsic Object" as shown in Figure 8 (b). Note that "Extrinsic Objects" point to the Repository items where their contents are stored. An OWL definition of an archetype is created from its ADL (Archetype Definition Language) [2] definition and is stored in the Repository. This OWL document gives the content of the archetype describing the constraints over reference information model classes.
- In order to establish the relationship with archetype "Extrinsic Objects" and the "archetype metadata ontology", an OWL instance of the "Archetype Metadata Ontology" is created which specifies the property values of an

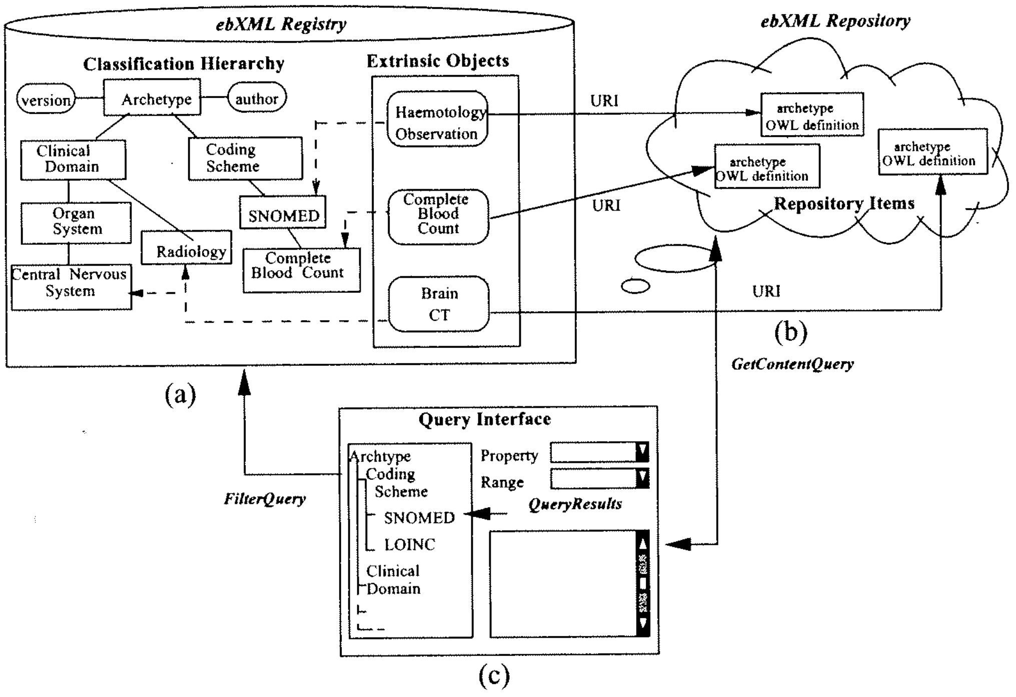


Figure 8 Querying Archetype Semantics in ebXML Registry

archetype according to this ontology. As an example, Figure 9 shows an "Archetype Metadata Ontology" instance that defines the "Complete Blood Count (CBC)" archetype metadata by providing the property values in "Archetype Metadata Ontology". Through the tool developed, while storing the "ExtrinsicObject" representing, for example the CBC archetype to the registry, the "Archetype Metadata Ontology" instance in Figure 9 is parsed and the relations between the "Extrinsic Object" and the "Classification Nodes" are created automatically in the Registry through "Classification Objects". In RIM, any RegistryObject may be classified using ClassificationSchemes and ClassificationNodes through "Classification Objects". In Figure 9, the CBC archetype is related with SNOMED "Complete blood count" term with "hasLinkTo" property. This relation is represented with the "Classification Object" of ebXML RIM as presented in Figure 10. Note that "CompleteBloodCount" class is a subclass of SNOMED.

After storing the archetype along with its semantic annotation to the ebXML Registry as presented in Figure 8(a), it becomes possible to query the archetypes according to their metadata. To facilitate the querying of the ebXML registry for novice users, we have developed a query tool with a GUI which on the left pane shows the classification hierarchy and allows users to formulate their queries by simply selecting the values automatically shown on the right pane according to the selections made on the left pane as shown in Figure 8(c). When a user selects values, Filter Queries are constructed for retrieving the ID's of the Extrinsic

```
<LabReport rdf:ID="LabReport_Instance"/> <Staff
rdf:ID="GokceBanuLaleci"/> <CompleteBloodCount
 rdf:ID="Snomed_Instance"/>
<Hemotology rdf:ID="Hematology_Instance"/>
<ClinicalInformationTemplate rdf:ID=
"ClinicalInformationTemplate_Instance"/> <Archtype
rdf:ID="CBC_Archetype"> <relatedTemplate rdf:resource=
"#ClinicalInformationTemplate_Instance"/> <relatedClinicalDomain>
    <Medicine rdf:ID="Medicine_Instance"/>
</relatedClinicalDomain> <isAPartOf
rdf:resource="#LabReport_Instance"/> <relatedTemplate>
    <AssesmentTemplate rdf:ID="AssesmentTemplate_Instance"/>
</relatedTemplate> <hasLinkTo rdf:resource="#Snomed_Instance"/>
<relatedClinicalDomain rdf:resource="#Hematology_Instance"/>
<relatedTemplate>
    <DiagnosticTestsTemplate rdf:ID=
    "DiagnosticTestsTemplate_Instance"/>
</relatedTemplate> <author rdf:resource="#GokceBanuLaleci"/>
<version rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>0.1</version>
<hasPurpose>
    <Clinical rdf:ID="Clinical_Instance"/>
</hasPurpose> <relatedClinicalDomain>
    <HemovascularSystem rdf:ID="HaemovascularSystem_Instance"/>
</relatedClinicalDomain> </Archtype>
```

Figure 9 "Complete Blood Count" Archetype Instance Definition

```
<ExtrinsicObjectid="CBCarchetypeInstance" mimeType="text/xml">
          <Name>
                <LocalizedString lang="en_US" value =
                "Complete Blood Count"/>
          </Name>
</ExtrinsicObject>
< Classification
classificationNode="CompleteBloodCount"
classifiedObject="CBCarchetypeInstance">
          <Slot name = 'type'>
                <ValueList>
                <Value>hasLinkTo</Value>
                </ValueList>
          </Slot>
</Classification>
```

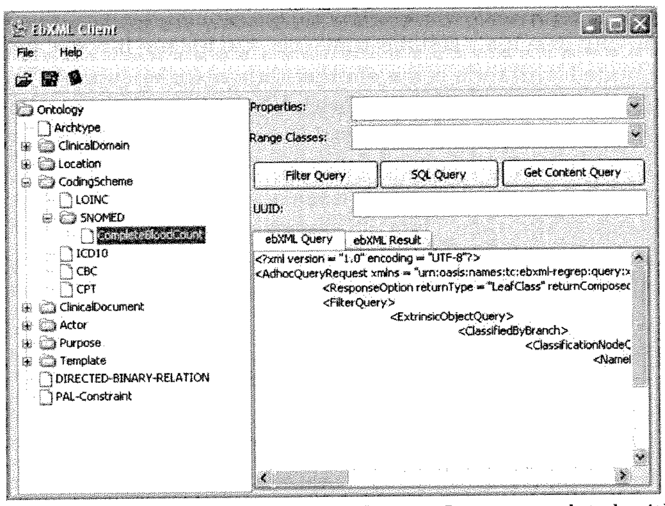
Figure 10 RIM Classification example

Objects, representing the Archetype Instances, according to the selected criteria. Then through these IDs ebXML "GetContent" queries are submitted for retrieving the repository items containing the archetype definition.

The ebXML semantic constructs can be used to query the archetypes through the GUI tool as follows:

- A user can search for all templates and archetypes that constrain to a particular code or coding scheme. For instance, a user can find all templates and archetypes that make reference to the SNOMED Complete Blood Count term, through the GUI tool as shown in Figure 11.

  When the SNOMED term is selected on the left pane, the FilterQuery in Figure 13 is generated to retrieve the ExtrinsicObjects IDs classified with the "CompleteBloodCount" ClassificationNode. In order to get the content of these extrinsic objects, the user can press the "Get Content Query" button presented in Figure 11. This will automatically generate the "ebXML



```
<?xml version = "1.0" encoding = "UTF-8"?>
<AdhocQueryRequest xmlns = "urn:oasis:names:tc:ebxml-regrep:query:x
        <ResponseOption returnType = "LeafClass" returnComposec
        <FilterQuery>
                <ExtrinsicObjectQuery>
                        <ClassifiedByBranch>
                                <ClassificationNodeC
                                        <Namel
```

Figure 11 The GUI for retrieving the archetype Instances related with CompleteBloodCount Concept

```
<GetContentRequest>
    <rim:ObjectRefList>
        <rim:ObjectRef id="urn:uuid:368661c9-b733-4c14-96a3
        -eabbdf36ff5b"/>
    </rim:ObjectRefList>
</GetContentRequest>
```

Figure 12 ebXML GetContentRequest

```
<FilterQuery>
   <ExtrinsicObjectQuery>
    <ClassifiedByBranch>
      <ClassificationNodeQuery>
        <NameBranch>
          <LocalizedStringFilter>
           <Clause>
            <SimpleClause leftArgument = "value">
              <StringClause stringPredicate =
              "Equal" >CompleteBloodCount</StringClause>
            </SimpleClause>
           </Clause>
          </LocalizedStringFilter>
        </NameBranch>
      </ClassificationNodeQuery>
    </ClassifiedByBranch>
   </ExtrinsicObjectQuery>
</FilterQuery>
```

Figure 13 ebXML Filter Query for Extrinsic Objects classified with CompleteBloodCount node

GetContentRequest" query given in Figure 12 to retrieve the OWL archetype definition from the Repository. The user can visualize the results of the query from the "ebXML Result" tab of the GUI given in Figure 11.

Note that there may be archetypes in the registry representing the same clinical concept but annotated with different clinical coding systems. Continuing with our example, assume that the "Complete Blood Count" archetype is annotated with term codes of other terminologies such as "Full Blood Count" of MedDRA [34] rather than with SNOMED "CompleteBloodCount" term. On the other hand, the user may be interested in a certain type of archetype independent of the coding system used to annotate it. In other words, we should be able to find the equivalency among the terms of different coding systems at run time. The US National Library of Medicine's Unified Medical Language System (UMLS) [42] provides a good resource for this purpose. UMLS is the official repository of many healthcare informatics' terminology standards. It contains information over one million biomedical concepts from more than one hundred controlled vocabularies and classifications (some in multiple languages) in the medical domain. It also provides the mapping between different terminologies. We have implemented a Web service, which takes a given coding term and its associated coding system and finds equivalent terms coded through other terminologies by using the UMLS.

Continuing with our example, we use this Web service for querying UMLS Metathesaurus for obtaining the synonyms of SNOMED "CompleteBloodCount" term automatically. This Web service returns:

- MedDRA - Full Blood Count
- Read Codes Full Blood Count

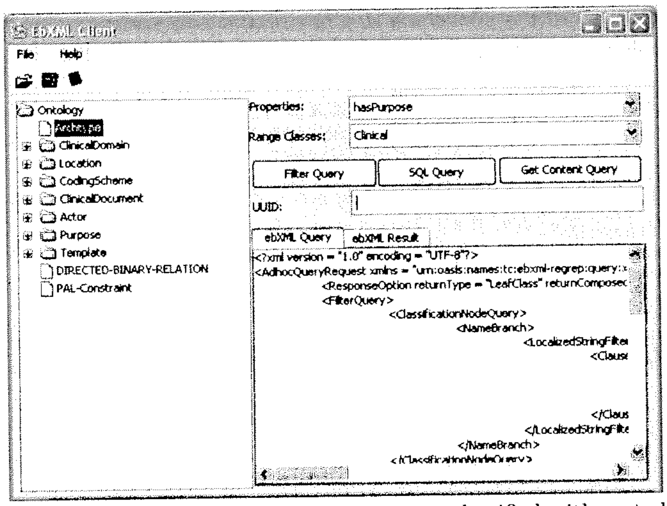
After obtaining the synonyms of SNOMED "CompleteBloodCount" term, the registry is queried for other ExtrinsicObjects, i.e. other archetypes, which are classified with the ClassificationNodes of these synonyms. These queries are expressed as shown in Figure 13.

After obtaining the ID's of the Extrinsic Objects, the archetype definitions can be retrieved from the Repository through "ebXML GetContentRequest" queries similar to the one depicted in Figure 12.

Note that, a healthcare institute may be using some local coding schemes that has not been defined in UMLS. In such a case, these schemas can be stored in the ebXML registry by defining the equivalences of their terms with other Clinical Coding Schema terms through the ebXML "EquivalentTo" association. Then by using the "findEquivalentInstances" stored procedure given in Section 3.3, it is possible to retrieve all the archetype instances of a Clinical coding scheme term as well as the archetype instances of all terms equivalent to this term.

- A user can search against any or all of the template and archetype metadata fields when looking for a template or an archetype.

  For example, it is possible to query the ebXML registry for "archetypes" that has been linked to "Clinical" class with "hasPurpose" property. As shown in Figure 14, when "Archetype" is selected on the left pane, all its properties are shown to the user in the related combo box so that the user can make a choice. The range class of the selected property can again be chosen from the ontology presented in the left pane.



Properties: hasPurpose

Range Classes: Clinical

UUID:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<AdhocQueryRequest xmlns = "urn:oasis:names:tc:ebxml-regrep:query:x
        <ResponseOption returnType = "LeafClass" returnComposec
        <FilterQuery>
                <ClassificationNodeQuery>
                        <NameBranch>
                                <LocalizedStringFilter
                                        <Clause

                                        </Claus
                                </LocalizedStringFilte
                        </NameBranch>
                </ClassificationNodeQuery>
```

Figure 14 The GUI for discovering the archetypes classified with metadata ontology

This GUI generates the necessary queries to the registry as follows: First the "ebXML IDs of Extrinsic Objects" of the archetypes annotated with the specified property are retrieved. In order to achieve this, two consecutive Filter queries (Figure 15) or one SQLQuery (Figure 16) are used by the system. In the first FilterQuery, the ID of the "Clinical" ClassificationNode is obtained and in the second one the Classifications that have a "type" Slot with value "hasPurpose" and that are bound to the "Clinical" ClassificationNode are retrieved. The result of the second query contains the ID's of the ExtrinsicObjects. These ID's can be used to retrieve the contents of the archetypes through "ebXML GetContentRequest" queries.

6 How to retrieve archetype data from medical information systems?

When we want to retrieve the associated data out of individual patient records conforming to the templates and archetypes, the functional interoperability issue also needs to be addressed.

ebXML provides functional interoperability through its messaging system which gives the specification of a standard way to exchange messages between organizations [16]. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP. It extends the base Simple Object Access Protocol (SOAP) [40] with MIME Attachments [41] for binding. All the interactions with the ebXML Registry as well as the interactions in a business process are specified to be handled through ebXML messages. There are implementations of ebXML messaging available both publicly and commercially.

Later, however ebXML also started providing registry support for Web services [17] which is another standard to provide functional interoperability. Web services are a set of related application functions that can be programmatically invoked over

```
<FilterQuery>
    <ClassificationNodeQuery>
        <NameBranch>
            <LocalizedStringFilter>
                <Clause>
                    <SimpleClause leftArgument = "value">
                        <StringClause stringPredicate =
                         "Equal">Clinical</StringClause>
                    </SimpleClause>
                </Clause>
            </LocalizedStringFilter>
        </NameBranch>
    </ClassificationNodeQuery>
</FilterQuery>

<FilterQuery>
    <ClassificationQuery>
        <SlotBranch>
            <SlotFilter>
                <Clause>
                    <SimpleClause leftArgument = "name">
                        <StringClause stringPredicate =
                        "Equal">type</StringClause>
                    </SimpleClause>
                </Clause>
            </SlotFilter>
            <SlotValueFilter>
                <Clause>
                    <SimpleClause leftArgument = "value">
                        <StringClause stringPredicate =
                         "Contains">hasPurpose</StringClause>
                    </SimpleClause>
                </Clause>
            </SlotValueFilter>
        </SlotBranch>
        <ClassificationFilter>
            <Clause>
                <SimpleClause leftArgument = "classificationnode">
                    <StringClause stringPredicate =
                    "Equal">urn:uuid:ef039f8f-0170-42a6-a329
                    -bfb40c8fe3a9</StringClause>
                </SimpleClause>
            </Clause>
        </ClassificationFilter>
    </ClassificationQuery>
</FilterQuery>
```

Figure 15 Two consecutive ebXML Filter Query

```
<SQLQuery>
    SELECT * FROM extrinsicobject E, classification CL1, slot S
    WHERE S.name LIKE 'type' AND S.value LIKE 'hasPurpose' AND
     S.parent = CL1.id AND CL1.classifiedobject = E.id
     AND CL1.classificationnode IN (
        SELECT C.id FROM Name N, ClassificationNode C
        WHERE C.id = N.parent AND N.value LIKE 'Clinical')
</SQLQuery>
```

Figure 16 ebXML SQL Query

the Internet. The information that an application must have in order to programmatically invoke a Web service is given by a Web Services Description Language (WSDL) [49] document. WSDL of a Web service, which defines its interface, is its established public contract with the outside world. The network protocol used is usually HTTP and binding is SOAP [40]. It should be noted that Web services not only provide synchronous invocation but also enable message exchange through asynchronous invocation.

We use Web services for the functional interoperability layer for the following reasons:

- Web services describe their interfaces through WSDL which gives a machine processable interface definition. Furthermore, WSDL has become a stable definition and there are several tools to automate the construction of an interface defined in WSDL such as IBM Web Services Toolkit [30], or Java Web Services Developer Pack [32].
- Functional interoperability standards, like Web services, need to improve various aspects of the usage scenarios. Several standardization initiatives are under way for providing security [47], privacy [46], transaction support [48], and reliability of Web services [45]. Although ebXML Messaging Services Specification Version 3 [15], also considers some of these issues, it has not been finalized yet.
- Finally, all the application servers support Web services such as BEA WebLogic [4], IBM WebSphere [29] and Oracle Application Server [38].

Web services retrieving data out of individual patient records conforming to the templates and archetypes, can be stored to ebXML registry as "Service" objects. It is possible to annotate such web services with the "Archetype Metadata Ontology" stored in ebXML registy. In this way, the discovery of these Web services through archetype semantics are facilitated.

When retrieving the associated information out of individual patient records conforming to the templates and archetypes through Web services, deciding on the granularity of Web service is important since this effects the service reusability and interoperability with other healthcare standards.

Among the archetype definitions, the ones that contain "elementary" information should be retrieved by "elementary" Web services, whereas composite archetypes should be retrieved by composing elementary Web services through workflow technology. Note that there could be composition relationship between archetypes when at some node in an archetype, a new archetype would occur, rather than a continuation of the constraints in the current archetype. This composition on the other hand can be handled with Web service composition techniques as explained in Section 7.

7 Composing Archetypes through Web service Composition

Annotating archetypes with the "Archetype Metadata Ontology" and facilitating their discovery from the ebXML registry enables us to create templates by dynamically composing semantically annotated archetypes. In this section, we present a Composition tool through which given the template's semantic definition, it becomes possible to construct the actual template containing the OWL descriptions of the involved archetypes and to execute it through Web services retrieving the clinical content.

For this purpose the Composition tool allows:

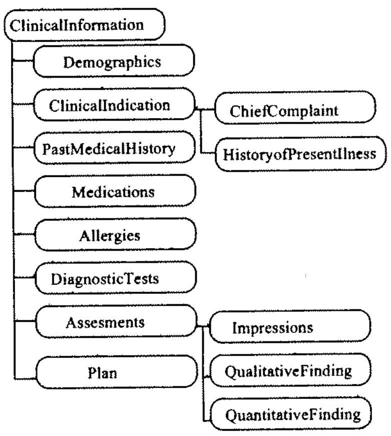


Figure 17 The Clinical Information Template

- To create a template definition and to retrieve the constituent archetype OWL definitions by discovering them from the ebXML registry,
- To create a composite Web service to retrieve the corresponding data conforming to the template definition from the underlying medical information systems. Here the Web services are also discovered from the ebXML registry by using the related semantics.

In the template semantic definition, template designers refer to the "Archetype Metadata Ontology" to indicate what kind of elementary archetypes constitute the template, rather than explicitly constructing the template definition by including the specific archetype OWL definitions. Through the tool developed, the template is composed by retrieving the archetype OWL definitions from the Repository that has been annotated by the related "Archetype Metadata Ontology nodes". The resulting OWL template definition can also be stored in the ebXML Registry by annotating it with the proper nodes in the "Archetype Metadata Ontology".

As an example, consider the "Clinical Information Template" presented in Figure 17. In its semantic definition, the archetypes which are the parts of this template are presented by refering to the "Archetype Metadata Ontology nodes". For instance, the "Allergies" component is annotated with "Clinical Information Template" node presented in Figure 2 and "Allergies and Immunology" Clinical Domain as presented in Figure 7, and "Medications" components is annotated with "Medication" Document Type presented in Figure 5.

In order to compose this template, these archetype instances are discovered from the ebXML Registry. The Composition tool provides a "Generic Archetype Proxy" for this purpose. The steps necessary to retrieve archetype instances are as follows (Figure 18):

- The "Generic Archetype Proxy" parses the semantic annotations presented in the template semantic definition.
- Exploiting these annotations, the "Generic Archetype Proxy" constructs the

relevant ebXML Filter queries to retrieve the Extrinsic objects in the ebXML registry representing the archetypes as shown in Figure 18 2.a.

- Then through "GetContent" queries the content of these archetypes are retrieved from the Repository (Figure 18 2.a.).
- If further constraints have been defined in the Template definition (such as it should use "SNOMED" Coding Scheme), the "Constraint Checker" component checks the suitability of the archetype instance with this template definition.
- The template definition is created in terms of the discovered archetype OWL definitions through the "Component Binder" (Figure 18 4.a).

For creating the composite Web service definition retrieving the medical data conforming to this template, there is a need to discover Web services providing the archetype data. In an ebXML registry, Web services can also be annotated with "Archetype Metadata Ontology" [12]. In this way it becomes possible to discover the Web services of medical institutes exchanging the data conforming to a specific archetype definition from the ebXML registry. As presented in Figure 1, this is achieved through the following steps:

- Web services are represented as "Service" entities in ebXML registry. Each Service object has a "specificationLink" attribute pointing to an "ExtrinsicObject". Through this "Extrinsic Object", the URI of the WSDL file of the Web service is referred. In Figure 1, the "specificationLink" of the "MyCBC-Service" points to the "MyCBC WSDL" extrinsic object, which in turn refers to the actual "Service WSDL" file stored in the Repository.
- In order to relate the "Service" object with an archetype, services are also annotated with the "ClassificationNodes" representing the "Archetype Metadata Ontology Nodes" through "Classification Objects". In Figure 1, "MyCBCService" is categorized with the "Complete Blood Count" "ClassificationNode".

In order to execute a composite Web service definition for retrieving data defined in a template, the Web services retrieving archetype instances are discovered from the ebXML Registry. Through the "Generic Archetype Proxy" presented in Figure 18 2.b, the Template semantic definition is parsed, and necessary ebXML Filter queries are prepared in order to retrieve the Service objects related with the "Archetype Metadata Ontology Nodes". The "Extrinsic Object" with which the "Service" is associated, is presented in the "Service Object" received as a response to these Filter queries. Then through "GetContent" queries, the WSDL files of the Web services can be retrieved from the Repository. Finally in the "Component Binder" component, the complex business process definition in BPEL4WS [8] is created as presented in Figure 18 4.b.

8 Conclusions and Future Work

Interoperability is a challenging problem in the healthcare domain, and archetypes are a promising approach for tackling this problem. However for archetypes to be

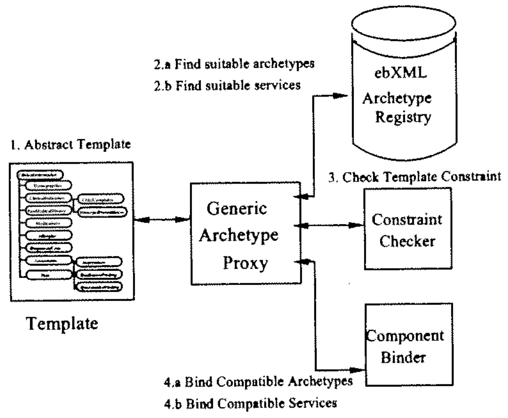

2.a Find suitable archetypes
2.b Find suitable services
ebXML Archetype Registry
1. Abstract Template
3. Check Template Constraint
Generic Archetype Proxy
Constraint Checker
Template
Component Binder
4.a Bind Compatible Archetypes
4.b Bind Compatible Services

Figure 18 Dynamic Archetype Binding

used as a shared model of domain concepts, there is a strong need for mechanisms for sharing archetypes through an archetype server, discovering archetypes through their semantics, facilitating template composition from archetypes and retrieval of corresponding data from the underlying medical information systems.

In this paper, we provide guidelines on how ebXML Registries, through their semantic constructs, can be exploited as an efficient medium for annotating, storing, discovering and retrieving archetypes. We also present how archetype data can be retrieved from proprietary clinical information systems by using ebXML Web services. This work is carried out within the scope of Artemis project [3] which aims to provide interoperability in the healthcare domain through semantically enriched Web services.

As already mentioned archetype based interoperability necessitates "harmonised" information model to be adopted. However, as an alternate solution, the reference models of the standardization bodies like openEHR, CEN TC/251, and HL7 can be represented through an ontology language like OWL. As a future work, we plan to define the archetypes constraining these reference models in OWL, and provide the mapping between these reference models through ontology mapping.

References and Notes

1 Aden, T, Eichelberg, M., Artemis Deliverable D3.1.1.4: Review of the State-of-the-Art-Healthcare standards: HL7, GEHR and CEN TC251 ENV 13606, CEN TC251 prEN 13606-1, http://www.srdc.metu.edu.tr/webpage/projects/artemis/calendar.php.

2 Archetype Definition Language (ADL) 1.2 draft, http://www.openehr.org/drafts/ADL-1_2_draftF.pdf

3 Artemis Project, http://www.srdc.metu.edu.tr/webpage/projects/artemis

4 BEA WebLogic Platform, http://e-docs.bea.com/platform/docs81/index.html

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

6 Beale, T., Heard, S.,"Archetype Definitions and Principles", http://www.openehr.-org/repositories/spec-dev/latest/publishing/architecture/archetypes/principles/-REV_HIST.html

7 Beale, T., Heard, S.,"The OpenEHR archetype system", http://www.openehr.-org/repositories/spec-dev/latest/publishing/architecture/archetypes/system/-REV_HIST.html

8 Business Process Execution Language for Web Services Version 1.1 , http://www-128.ibm.com/developerworks/library/ws-bpel/

9 Complete Blood Count Archetype ADL Definition, http://www.openehr.org/-repositories/archetype-dev/adl_1.1/adl/archetypes/openehr/ehr/entry/openehr-ehr-observation.haematology-cbc.draft.adl.html

10 CEN TC/251 (European Standardization of Health Informatics) prEN13606, Electronic Health Record Communication, http://www.centc251.org/

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

12 Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004.

13 A. Dogac, Y. Kabak, G. Laleci, C. Mattocks, F. Najmi, J. Pollock, "Enhancing ebXML Registries to Make them OWL Aware", Submitted to the Distributed and Parallel Databases Journal, Kluwer Academic Publishers. http://www.srdc.metu.edu.tr/-webpage/publications/2004/_DAPD_ebXML-OWL.pdf.

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

15 ebXML version 3, http://www.oasis-open.org/committees/download.php/4383/-ebMSv3FeaturePreview

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

17 ebXML Registry Information Model v2.0, http://www.ebxml.org/specs/ebrim2.pdf

18 ebXML Registry Information Model v2.5, http://www.oasis-open.org/committees/-regrep/documents/2.5/specs/ebRIM.pdf

19 ebXML Registry Services Specification v2.5, http://www.oasis-open.org/committees/-regrep/documents/2.5/specs/ebRIM.pdf

20 Elkin, P., Kernberg, M., "HL7 Template and Archetype Architecture", HL7 Template Special Interest Group, Jan. 2004.

21 Fensel, D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.

22 Heard, S., Beale, T.,Mori, A.R., Pishec, O.,"Templates and Archetypes: how do we know what we are talking about" http://www.openehr.org/downloads/archetypes/templates_-and_archetypes.pdf

23 IHE IT Infrastructure Integration Profiles, http://www.rsna.org/IHE/tf/ihe_tf_index.-shtml

24 IHE IT Infrastructure Technical Framework, Cross-Enterprise Clinical Documents Sharing (XDS), by the IHE ITI Technical Committee, Version 3.0, April 26, 2004.

25 Health Level 7 (HL7), http://www.hl7.org

26 HL7 Clinical Document Architecture, Release 2.0. PDF format. Version: August 30, 2004, http:/xml.coverpages.org/CDA-20040830v3.pdf

27 HL7 Reference Information Model, http://www.hl7.org/library/data-model/RIM/modelpage_non.htm

28 HL7 Version 3 Message Development Framework, http://www.hl7.org/library/mdf99/mdf99.pdf

29 IBM WebSphere Application Server Technology for Developers V6, http://www-106.-ibm.com/developerworks/websphere/downloads/WASTD6Support.html

30 IBM Web Services Toolkit, http://www.alphaworks.ibm.com/tech/webservicestoolkit

31 ISO TC/215, International Organization for Standardization, Health Informatics, ISO TS 18308:2003, secure.cihi.ca/cihiweb/en/downloads/infostand_ihisd_isowg1_mtg-_denoct_contextdraft.pdf

32 Java Web Services Developer Pack (Java WSDP), http://java.sun.com/webservices/-jwsdp/index.jsp

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

34 MedDRA: Medical Dictionary for Regulatory Activities, http://www.fda.gov/medwatch/report/meddra.htm

35 Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.

36 OASIS: Organization for the Advancement of Structured Information Standards, 1998, http://www.oasis-open.org/cover/siteIndex.html.

37 openEHR Community, http://www.openehr.org/

38 Oracle Application Server Web Services Tutorials, http://www.oracle.com/technology/-tech/webservices/htdocs/series/index.html

39 SNOMED Clinical Terms, http://www.snomed.org/snomedct_txt.html

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

41 SOAP Messages with Attachments, http://www.w3.org/TR/SOAP-attachments.

42 Unified Medical Language System (UMLS), http://www.nlm.nih.gov/research/umls/

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

44 Web Ontology Language OWL, http://www.w3.org/TR/owl-features/

45 Web Services Reliability (WS-Reliability), http://developers.sun.com/sw/platform/-technologies/ws-reliability.html

46 Handling Privacy In WSDL 2.0, http://www.w3.org/TeamSubmission/2004/SUBM-p3p-wsdl-20040213/

47 Web Services Security (WS-Security), http://www-106.ibm.com/developerworks/-webservices/library/ws-secure/

48 Web Services Transaction (WS-Transaction), http://www-106.ibm.com/-developerworks/webservices/library/ws-transpec/

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

Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain ★

A. Dogac, G. Laleci, S. Kirbas, Y. Kabak, S. Sinir, A. Yildiz

Software Research and Development Center

Middle East Technical University (METU)

06531 Ankara Turkiye

email: asuman@srdc.metu.edu.tr

Abstract

An essential element in defining the semantic of Web services is the domain knowledge. Medical informatics is one of the few domains to have considerable domain knowledge exposed through standards. These standards offer significant value in terms of expressing the semantic of Web services in the healthcare domain.

In this paper, we describe the architecture of the Artemis project, which exploits ontologies based on the domain knowledge exposed by the healthcare information standards through standard bodies like HL7, CEN TC251 , ISO TC215 and GEHR.

Artemis Web service architecture does not propose globally agreed ontologies; rather healthcare institutes reconcile their semantic differences through a mediator component. The mediator component uses ontologies based on prominent healthcare standards as references to facilitate semantic mediation among involved institutes. Mediators have a P2P communication architecture to provide scalability and to facilitate the discovery of other mediators. The overall architecture of the system is adapted from the Semantic Web enabled Web services proposal.

1 Introduction

Most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. To complicate the matters worse, a patient's health information may be spread out over a number of different institutes which do not interoperate. This makes it very difficult for clinicians to capture a complete clinical history of a patient.

On the other hand, the Web services model provides the healthcare industry with an ideal platform to achieve the difficult interoperability problems. Web services are designed to wrap and expose existing resources and provide interoperability among diverse applications.

Introducing Web services to the healthcare domain brings many advantages:

- It becomes possible to provide the interoperability of medical information systems through standardizing the access to data through WSDL [36] and SOAP [33] rather than standardizing documentation of electronic health records.
- Medical information systems suffer from proliferation of standards to represent the same data. Web services allow for seamless integration of disparate applications representing different and, at times, competing standards.
- Web services will extend the healthcare enterprises by making their own services available to others.
- Web services will extend the life of the existing software by exposing previously proprietary functions as Web services.

However it has been generally agreed that Web services offer limited use unless their semantics are properly described and exploited [24–26,28].

⋆ This work is supported by the European Commission through IST-1-002103-STP Artemis project and in part by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 102E035

Semantic Web Enabled Web Services (SWWS) [5] architecture considers semantics as a vertical layer that may be exploited by the horizontal layers of Web service stack such as service description (including the documents exchanged), publishing, discovery as well as service flow and composition as shown in Figure 1 [5].

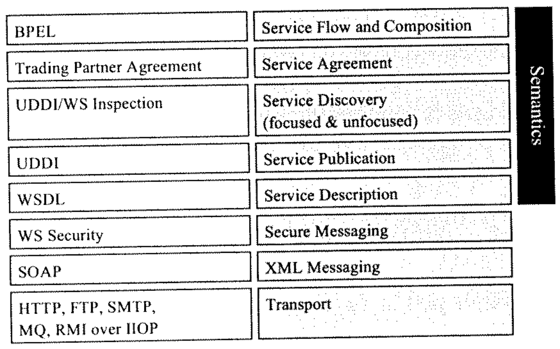


Fig. 1. Web Service Stack and Semantic

Another essential element in defining the semantic of Web services is the domain knowledge. The healthcare information standards through standard bodies like HL7 [18], CEN TC251 [6] , ISO TC215 [21] and GEHR [17] expose considerable domain knowledge through classifications, methodologies, terminologies, and controlled vocabularies.

In this paper, we present the design and implementation of a semantically enriched Web service based interoperability platform for the healthcare domain which is being developed within the scope of the Artemis project [2]. The main contributions of the architecture are as follows:

- HL7 has categorized the events in healthcare domain by considering service functionality which reflects the business logic in this domain. We propose to use this classification as a basis for defining the service action semantics through a *Service Functionality Ontology*. In the Web services protocol stack, this corresponds to the semantics of service descriptions.
- It is also necessary to define the semantics of documents exchanged through Web services. When ontologies are available, then documents can refer to ontology concepts, hence

allowing for the semantic mediation of the concepts in the documents.

Electronic healthcare record (EHR) based standards like HL7 CDA (Clinical Document Architecture) [12], GOM (GEHR Object Model) [3] and CEN TC251's ENV 13606 [6] define meaningful components of EHR so that when transferred, the receiving party can understand the record content better.

We propose to organize the "meaningful components" defined by these standards into ontologies and use such ontologies in associating semantics with the documents exchanged between the healthcare institutes.

- Although we propose to develop ontologies based on the prominent healthcare standards, the ontologies we are proposing is just to facilitate ontology mediation. In other words, we do not find it realistic to expect healthcare institutes to conform to one global ontology. In Artemis architecture, the healthcare institutes can develop their own ontologies. However, when these ontologies are based on standards developed by the healthcare standardization bodies like CEN TC251, ISO TC215, GEHR or HL7, we show that ontology mappings are facilitated to a great extend through semantic mediation.

  The mediator architecture in Artemis is based on a peer-to-peer infrastructure to provide scalability and to facilitate the discovery of other mediators.

- Although classifying the Web Services through the "semantic category" of the data they are providing facilitates the discovery of the services fetching a specific part of EHR data, it may not always be possible to find a service delivering exactly the data requested. For example, a healthcare institute may be providing the diagnosis information as a part of another clinical concept. This may necessitate more complex aggregations of Web Services. We address how complex aggregation of Web services can be handled by taking advantage of the ontology mappings.

The paper is organized as follows: In Section 2, we describe how the semantics exposed by the healthcare standards can be taken advantage of in developing a Web service technology framework for healthcare domain. We introduce *Service Functionality Ontology, Service Message Ontology* and use a MAFRA [23] based ontology mapping mechanism. How to compose complex services from elementary services using the semantics and how to aggregate services are also discussed in this section. In Section 3 we present the system architecture and the Artemis mediator component. Artemis builds upon the work of several others. In the Section 4, we present the previous work we have benefited. Finally Section 5 concludes the paper and discusses future work.

2 Exploiting Web Service Technology in Healthcare Informatics

Medicine is one of the few domains to have extensive domain knowledge defined through standards. Some of the domain knowledge exists in "controlled vocabularies", or "terminologies". Some vocabularies are rich semantic nets, such as SNOMED-CT [32] while others such as ICD-10 (International Statistical Classification of Diseases and Related Health Problems) [20] is little more than lexicons of terms. However, in addition to such vocabularies and taxonomies, there are standards that expose the business logic in the healthcare domain such as HL7 [18] and Electronic Healthcare Record based standards such as CEN TC251 [6], ISO TC215 [21] and GEHR [17] which define and classify clinical concepts. These standards offer significant value in developing ontologies to express the semantics of Web services.

The semantics is necessary in medical Web services in the following respects:

- First, in order to facilitate the discovery of the Web services, there is a need for an ontology to describe service functionality in the healthcare domain.

  For example, when a Web service instance, say "HastaKabul" is annotated with the "AdmitPatient" node of such an ontology, its operational meaning becomes clear that this service can be used in admitting patients to a hospital.

- Service functionality semantics is not enough; in real life medical information services, there can be quite complex service parameters and therefore both the semantics and the structure of the message parameters are also necessary to decipher them at the receiving end.

- As already noted, it is not realistic to expect global ontologies, rather it is possible to have more than one ontology to express the similar concepts. This is especially true for the medical information systems: the EHR based standards like CEN TC251 and GEHR use different terminologies for similar concepts.

  However, given these standards, it is also not realistic to ignore all these efforts and develop brand new standards. Therefore, it is reasonable to expect healthcare institutes to develop their own ontologies based on the concepts provided by the existing healthcare information standards.

  On the other hand, it is possible to specify the ontology mappings between existing standards. Such mappings make it possible to facilitate the mediation between healthcare institutes' own ontologies as long as they make use of ontologies based on these standards.

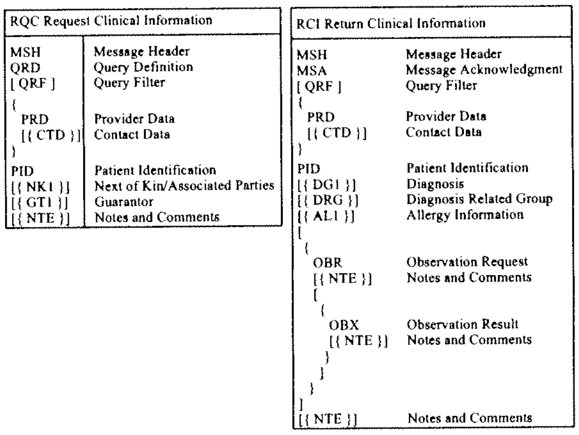
- The semantic constructs developed must be integrated with the service registries which provide the basic mechanisms for service discovery.

There are basically two different approaches to healthcare standardization efforts: The first approach is message based such as HL7 [18]; the other is Electronic Health Care Record (EHR) based such as CEN ENV 13606 [6], and GEHR [17]. In the following sections, we describe how these standards can be exploited in developing semantic based healthcare Web services.

2.1 HL7 and Web services

The primary goal of HL7 is to provide standards for the exchange of data among healthcare computer applications. The standard is developed with the assumption that an event in the healthcare world, called the *trigger event*, causes exchange of messages between a pair of applications. When an event occurs in an HL7 compliant system, an HL7 message is prepared by collecting the necessary data from the underlying systems and it is passed to the requestor, usually as an EDI message. For example, the trigger event can occur when a patient is admitted and this may cause the data about that patient to be collected and sent to a number of other systems.

Since HL7 defines message based events, one might think that these events can directly be mapped into Web services. However, this may result in several inefficiencies. The input and output messages defined for HL7 events are usually very complex containing innumerous segments of different types and optionality. Furthermore, all the semantics about the business logic and the document structure are hard coded in the message. This implies that, the party invoking the Web service must be HL7 compliant to make any sense of the content of the output parameter(s) returned by the service.



RQC Request Clinical Information	
MSH	Message Header
QRD	Query Definition
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ NK1 }]	Next of Kin/Associated Parties
[{ GT1 }]	Guarantor
[{ NTE }]	Notes and Comments

RCI Return Clinical Information	
MSH	Message Header
MSA	Message Acknowledgment
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ DG1 }]	Diagnosis
[{ DRG }]	Diagnosis Related Group
[{ AL1 }]	Allergy Information
[	
{	
OBR	Observation Request
[{ NTE }]	Notes and Comments
[	
{	
OBX	Observation Result
[{ NTE }]	Notes and Comments
}	
]	
}	
]	
[{ NTE }]	Notes and Comments

Fig. 2. The Structures of the RQC/RCI messages for the HL7 event I05

Note further that some of the information contained in an HL7 message may be coming from different systems either proprietary or complying to different standards. For example the event I05 in HL7 is used to pass the clinical patient information given patient identification information. Clinical information refers to the data contained in a patient record such as problem lists, lab results, current medications, family history, etc. [19]. The input and output messages of I05 are shown in Figure 2. All or some of this data may be coming from different systems that do not interoperate. This in turn, creates the need to retrieve these partial results probably through finer granularity Web services. Hence, in Web services terminology, HL7 events correspond to "Composite services", whereas more elementary services are needed. Deciding upon the "elementary" service granularity is important since this effects the service reusability and interoperability with other healthcare standards.

In order to define the granularity of Web services, we refer to Electronic Healthcare Record (EHR) based standards from major standard bodies like CEN and GEHR. These standards define metadata about EHR through "meaningful components".

When a Web service is designed to retrieve a fine granularity "meaningful component" of an EHR, it can be semantically annotated as such. In other words, we propose to annotate the semantic of fine granularity Web services through the semantic of the messages that they carry. This provides the following benefits:

- Its semantic can be mapped between different EHR standards to provide for interoperability. For example, a Web service retrieving "Allergy Information" can be semantically annotated as "AL1" in HL7. Then a CEN's ENV 13606 compliant system can understand the semantics of this service through an ontology mapping indicating that "AL1" in HL7 corresponds to "DF03" in a CEN's ENV 13606 compliant system.
- And its reusability is improved; it can not only be invoked by other applications which need only that piece of data but also be used as a component of a larger composite service. For example a service retrieving "Allergy Information" can be a part of a composite service retrieving the whole clinical information about a patient.

As a summary, there is a need for a *Service Functionality Ontology* to classify coarse-grained Web services in healthcare domain and also for a *Service Message Ontology* to annotate finer granularity services retrieving meaningful EHR components. These issues are detailed in the following sections.

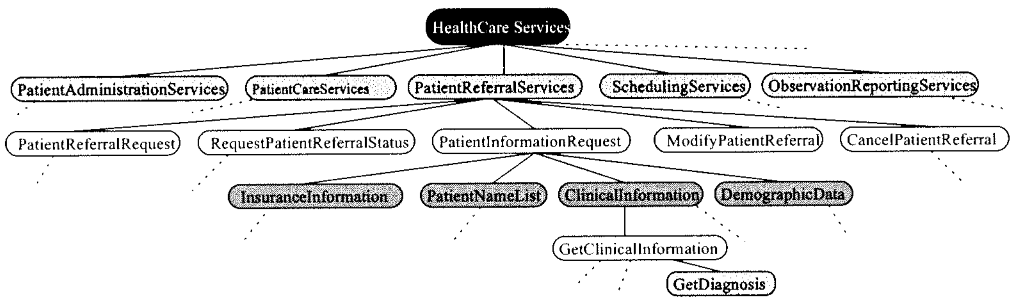


Fig. 3. A Service Functionality Ontology based on HL7

Since HL7 has already been through an effort of categorizing the events in healthcare domain considering service functionality, we propose to use this classification as a basis for a service functionality ontology.

The HL7 standard [18] groups the HL7 events into the following clusters: Patient Administration, Order Entry, Query, Financial Management, Observation Reporting, Master Files, Medical Records/Information Management, Scheduling, Patient Referral, and Patient Care. These clusters also have sub clusters. A partial Web service Functionality Ontology is given in Figure 3 based on HL7 events.

It should be noted that our aim is not to propose such an ontology but to show how such an ontology, once developed, can be used in semantic mediation. Furthermore, we assume that there could be more than one such ontologies which are mapped to one another through mapping rules.

When searching for the right Web services, consumers can consult this ontology to find out the semantics of the service they are looking for. Additionally, service discovery can be facilitated by associating the nodes of this ontology with the service instances explicitly. How this is achieved in UDDI and ebXML registries, is explained in Section 2.7.

2.3 Web Service Message Ontology

A Web service in the healthcare domain usually accesses or updates a part of an electronic healthcare record, that is, parts of the EHR constitute the service parameters. An electronic healthcare record may get very complex with data coming from diverse systems such as lab tests, diagnosis, prescription of drugs which may be in different formats.

As an example, consider the Web service given in Figure 8 Part (b). Although the semantic of action, the "Klinik_Bilgi_Saglayici" service is providing, is clear from the functionality ontology (i.e., it is retrieving clinical information about a patient), it is not clear what the content and format of service parameters like "PatientID" and "ClinicalInformation" are. To provide for interoperability, this additional message semantics is essential and we exploit the EHR based standards in this respect.

Electronic healthcare record (EHR) based standards like HL7 CDA (Clinical Document Architecture) [12], GOM (GEHR Object Model) [3] and CEN's ENV 13606 [6] aim to facilitate the interoperability between Medical Information Systems. However, they do not aim direct machine-to-machine interoperability. Therefore these standards do not prescribe a monolithic EHR architecture, rather they provide conceptual "building blocks" or "meaningful components" by which any clinical model can be represented within the standardized framework. This provides flexibility by allowing the same "building block" to be composed differently by two different institutes, which in turn results in different message structures. This necessitates structural and semantic mappings between the message components in order to automate their interoperation. It is possible to define "clinical concept" ontologies based on the "building blocks" of the EHRs, with ontology definition languages, such as OWL [35]. As an example, in Figure 4, two partial clinical concept ontologies are presented based on the "building blocks" of HL7 and ENV-13606.

In Artemis architecture, medical institutions provide Web Services for accessing the compo-

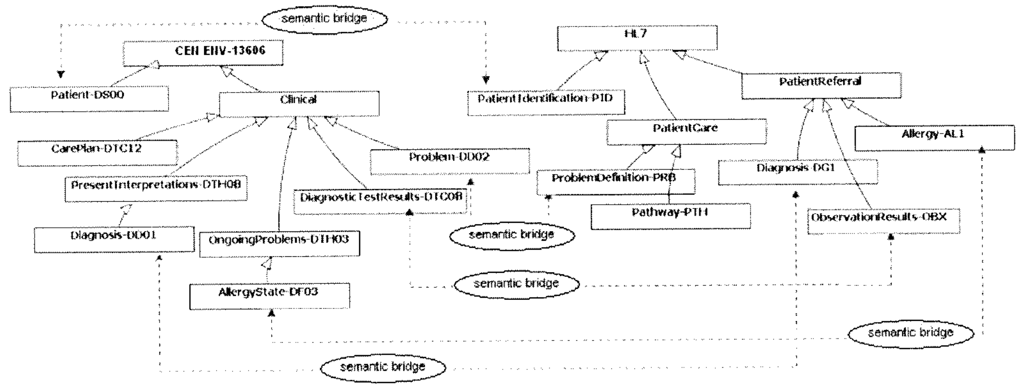


Fig. 4. CEN ENV-13606 and HL7 Clinical Concept Ontologies

nents of EHR with a granularity to retrieve the nodes (or the composition of the nodes) of the Clinical Concept Ontologies. The semantics of the service parameters are defined using "message ontologies" which are constructed by using these "clinical concept" ontologies. Once semantically marked up, these elementary Web services are classified under the *Service Functionality* ontology. For example, a Web service retrieving "Diagnosis" information can be classified under "GetClinicalInformation" node as shown in Figure 3.

The medical institutes can develop their own clinical concept ontologies to annotate their Web services. However if these ontologies are derived from the Clinical Concept Ontologies based on prominent healthcare standards like HL7, CEN TC251, ISO TC215 and GEHR, then the ontology mapping is facilitated.

2.4 Ontology Mapping

Although representation of the clinical concepts defined by different standardization efforts may result in disparate clinical ontologies initially; defining them through ontology languages opens up the way to mapping them one another through reasoning.

Consider the two partial clinical concept ontologies from HL7 and ENV-13606 presented in Figure 4. Once such clinical ontologies are defined, the mappings between them can be achieved using the available "Ontology Mappers" such as "MAFRA" [23]. MAFRA uses a mediator component that defines the relations and transformations between ontologies. Generally speaking, ontology mapping has three main dimensions: discovery, representation and execution. Discovery, which is the extraction of the semantic similarity relations between entities of the ontologies, is accomplished by using existing similarity measuring approaches, such as linguistic based algorithms [30]. In Artemis, ontologies are based on well-defined medical informatics standards which facilitate the discovery phase to a great extend.

For representing the similarities in a formal way, MAFRA provides a mediator component which is a meta-ontology called Semantic Bridge Ontology (SBO). Semantic Bridges in SBO encapsulate the required information to translate one source entity (concept, relation, property) to a target entity. Semantic Bridges provide mapping cardinality from 1:1 to m:n, and allow for complex structural mappings such as specialization, abstraction, composition and alternatives.

SBO also has concepts to specify conditions, transformation rules, and transformation functions (services) to be used during execution step. It is possible to specify conditions that need to be verified to execute the semantic bridges. Services are used to reference the resources that will be used to handle transformations (i.e. copy an attribute, split a string). SBO is represented in DAML-OIL in MAFRA.

MAFRA has two primitive semantic bridges: Concept Bridge, and Property Bridge. A Concept Bridge defines the semantic equivalence between two ontology classes. At execution step, an instance concept of the target ontology is created for each source concept when the two concepts are related via a concept bridge. In the same way a Property Bridge defines the equivalence between source and target properties.

Once the relationships between two ontologies are defined through "semantic bridges", the instances of source ontology can be transformed into target ontology instances by evaluating the "semantic bridges" at the execution step [23]. At this step, firstly, the instances of target ontology are created if the conditions of the related concept bridges evaluate to true. After all instances are created, property bridges are executed and the properties of target instances are set according to them. In Artemis project, this step is used for converting one healthcare institute's ontology (say, based on ENV-13606) into another (say, based on HL7) by obtaining the necessary "semantic bridges" from the mapping of original ENV-13606 and HL7 based ontologies.

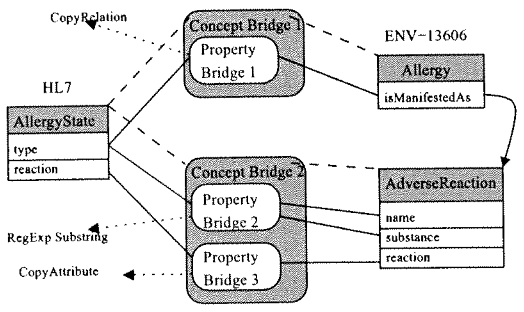


Fig. 5. An Example Mapping Using MAFRA Constructs

As an example, in Figure 5, a mapping using MAFRA constructs is illustrated. In this figure, the "Allergy State" concept of HL7 is mapped to the "Allergy" concept of ENV-13606 through semantic bridges. While a single class is used to represent the "Allergy State" in HL7, the same information is represented with two associated classes, namely "Allergy" and "Adverse Reaction" in ENV-13606. Hence to map these concepts, two "Concept Bridges" are constructed. The "type" attribute in "Allergy State" contains information about "name" and "substance" attributes of "Adverse Reaction". To represent this relation, the "Property Bridge 2" is added to the "Concept Bridge 2". In the execution step this mapping is handled with the help of "RegExp Substring" predefined service of MAFRA, which basically searches/splits a string via regular expressions. The "reaction" attributes in both ontologies which carry the same semantics, are directly mapped through the "Property Bridge 3". Finally, to express the semantic relation between the "Allergy" and "Adverse Reaction" "Property Bridge 1" is added to the "Concept Bridge 1".

When the semantics of finer granularity Web Services are defined in terms of the Clinical Concept Ontologies, it also becomes possible to determine how to compose a course-grained service from finer granularity services.

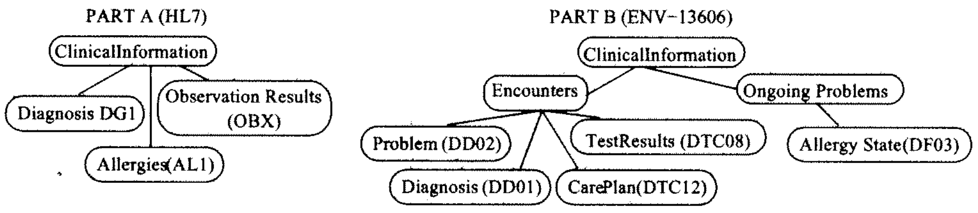


Fig. 6. Clinical Information Representation in two different Systems

To clarify this issue, consider Healthcare Institute A, which needs the Clinical Information of a patient stored in Healthcare Institute B. As previously stated Artemis gives the flexibility to the healthcare institutes to define their own clinical ontologies based on existing standards. Therefore Healthcare Institute A may define "Clinical Information" as presented in Figure 6 Part A, in terms of the Clinical Concepts defined by HL7 (Figure 4), and Healthcare Institute B may define the same concept, as depicted in Figure 6 Part B, in terms of the Clinical Concepts defined by ENV-13606. In fact these are parts of the "message ontologies" of these institutes, which are used in exchanging "Clinical Information". Notice that both the "building blocks" of these "message ontologies", and also their hierarchical structures are different. Therefore when Healthcare Institute A requests "Clinical Information" of a patient from Healthcare Institute B, there is a need for both structural and semantic transformation of the documents exchanged.

The semantic mappings between the concepts in these two "message ontologies" are handled by using Ontology Mappers such as MAFRA to process the "semantic bridges" defined between the "Clinical Concept Ontologies" (i.e. building blocks of these message ontologies) as shown in Figure 4.

If Healthcare Institute B is providing the Web Services for accessing the "Observation Results", "Allergies" and "Diagnoses", structural mappings are easily handled by discovering these Web Services and composing them to satisfy the request of Healthcare Institute A. Here we are assuming that the semantic mappings for the "Diagnosis" concept (as well as the "Allergies" and "Observation Problems") has already been defined through semantic bridges. Otherwise, the same decomposition process should be applied for the "Diagnosis" concept until finer granularity semantically agreed components are reached.

Since the Web services are annotated with Clinical Concept Ontologies, it is possible to identify the Web services providing the requested information such as "Diagnosis" from service registries. For this purpose the tModel keys associated with the nodes of Clinical Concept Ontologies are used to find related services in UDDI. In ebXML, a *Service Message* ontology exists (just like the *Service Functionality* ontology) and the related nodes of this ontology such as "Diagnosis:DD01" are used to find the requested services. The details of how this is achieved is presented in Section 2.7. In this way, the information requested in the ClinicalInformation record can be obtained as requested by the Healthcare Institute A from the Healthcare Institute B.

2.6 Semantic Aggregation of Medical Web Services

Although classifying the Web Services through the "semantic category" of the data they are retrieving facilitates the discovery of the services giving a specific part of EHR data, it may not always be possible to find a service delivering exactly the data requested. For example, a healthcare institute may be requesting "Diagnosis" information whereas the target institute may be providing the diagnosis information as a part of another clinical concept. This may necessitate more complex aggregations of Web Services (such as union, intersection). In other words when we try to compose a Web Service from fine granularity Web services according to the structure and the semantics of the composite Web service output parameter(s), we may not always find disjoint Web services to produce the requested output.

As an example consider the case where Healthcare Institute A is requesting Clinical information as shown in Figure 6 Part A but Healthcare Institute B provides Web Services only to retrieve "Encounter" and "Ongoing Problem" information of a patient (Figure 6). Given the semantic structure of "Encounters" and "Ongoing Problems" concepts, it is possible to construct the "Clinical Information" concept requested by Healthcare Institute A, through a set of *Semantic Aggregation Operators* (SAO). For example we can construct the "Clinical Information" concept of Healthcare Institute A (i.e. ClinicalInformation:A) as follows: ClinicalInformation:A = (ClinicalInformation:A $\cap_s$ Encounters:B) $\cup_s$ (ClinicalInformation:A $\cap_s$ OngoingProblems:B).

We call the Web Services constructed as "semantic aggregations" of other Web Services as *Virtual Web Services* (VWS). In other words, these virtual Web services are abstractions; they are neither instantiatable nor executable. Rather, they specify how to obtain the required output of a complex Web service from other Web services through *Semantic Aggregation Operators* (SAO).

In the example presented, the Healthcare Institute A uses a *Virtual Web Service* to retrieve the Clinical Information from the Healthcare Institute B.

2.6.1 Semantic Aggregation Operators

We propose a number of "Semantic Aggregation Operations" (SOA) in order to construct *Virtual Web Services* (VWS). These SOAs are as follows:

- $VWS1(\cup_s)VWS2$ *Semantic Union*: This operation can be used to construct a VWS which provides the semantically combined outputs of services VWS1 and VWS2. In other words the output of VWS includes the disjoint concepts provided by VWS1 and VWS2 and the semantically equivalent concepts provided by both only once.

Example: Semantic Union of the VWS whose output semantic is given in Figure 6 Part A with the VWS whose output semantic is given in the same figure Part B produces a VWS whose output semantic is same as the ontology in Figure 6 Part B. The input semantics of the resultant VWS is defined as the *Semantic Union* of the inputs of the involved Web services.

- $VWS1(\oplus_s)VWS2$ *Semantic Heaping*: This operation can be used to construct a VWS which provides the combined outputs of services VWS1 and VWS2 by collecting all the concepts that take place in both, and disregarding whether the concepts are semantically equivalent or not.

  Example: If we apply *Semantic Heaping* to two VWSs whose output semantics are given in Part A and Part B of Figure 6, the output semantics of the resultant VWS is as given in Figure 7. The input semantics of the resultant VWS is defined as the *Semantic Union* of the inputs of the involved Web services.

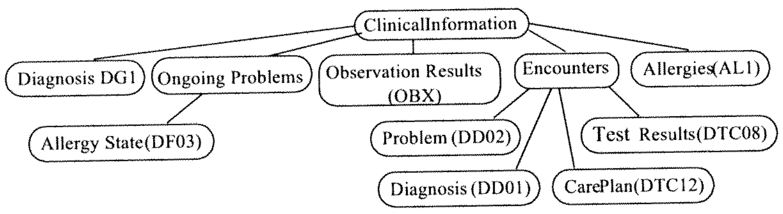


Fig. 7. Semantic Heaping Example

- $VWS1(\cap_s)VWS2$ *Semantic Intersection*: This operation can be used to construct a VWS which provides the semantically equivalent concepts provided by both VWS1 and VWS2.
- $VWS1(\ominus_s)VWS2$ *Semantic Difference*: This operation can be used to construct a VWS which gives the concepts provided by VWS1 excluding the semantically equivalent concepts provided by VWS2.

- $VWS1(\gg_s)VWS2$ *Semantic Contain*: This operation can be used to check whether the concepts provided by VWS1 is a superset of the concepts provided by VWS2.

Given these "semantic aggregation" operators, coarse grained Web Services can be composed from finer granularity services even when there is no finer granularity service retrieving exactly the requested data. For example, as shown in Figure 6, the Web Service providing "Encounters" Information of a patient is classified with the "clinical concepts" in its output such as Problem:DD02, TestResults:DTC08, Diagnoses:DD01, and CarePlan:DTC12 (Figure 6). Hence this Web Service is a candidate for aggregation in order to gather the data requested by Healthcare Institute A.

The results of these aggregations, i.e. Virtual Web Services are also re-usable components. They are inserted as instances into the *Service Functionality* ontology together with their descriptions. For instance, the virtual service retrieving Clinical Information of a patient in the running example, is stored as an instance of the "GetClinicalInformation" node of the *Service Functionality* ontology presented in Figure 3. Whenever these two hospitals interact again, these VWS definitions can be reused.

Providing such Virtual Web Services and creating a repository from these VWS, in the long run, may improve the interoperability of Medical Information Systems. Although these VWS may seem as bilateral agreements between institutes, they can be used to create a wider community through transitive agreements as discussed in [1].

2.7 Relating Web Service Ontologies with Web Service Registries

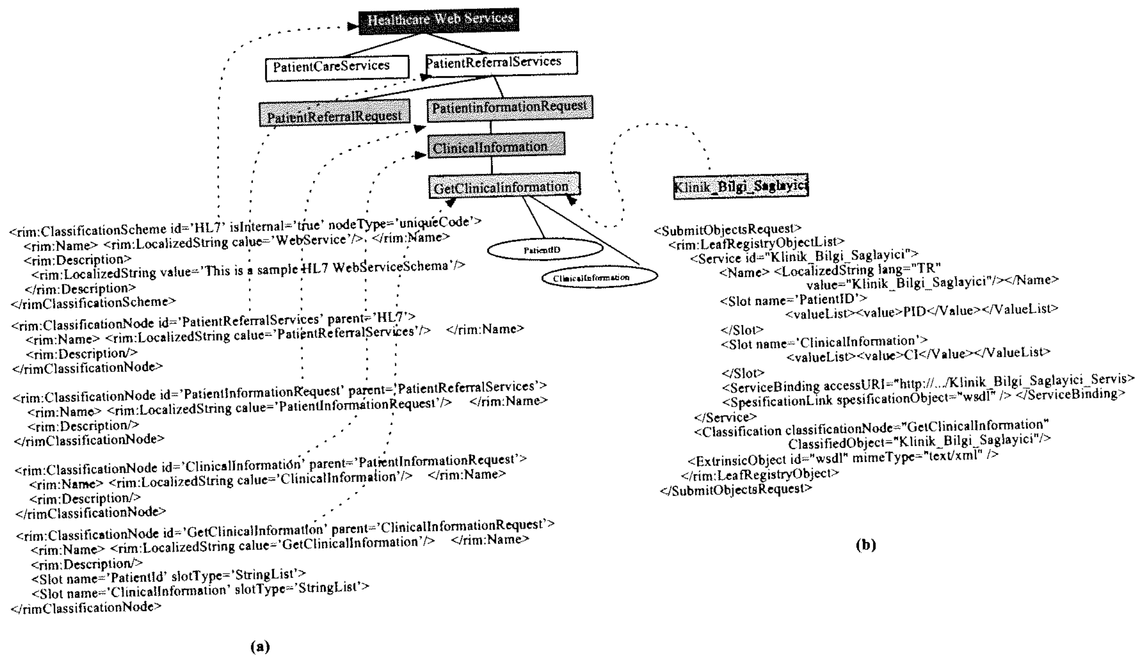
Once the semantic of Web services are specified, it is necessary to relate this with the services advertised in service registries.

There are two key issues in this process: the first one is where to store the ontologies. UDDI does not provide a mechanism to store an ontology internal to the registry. ebXML, on the other hand, through its classification hierarchy mechanism allows domain specific ontologies to be stored in the registries. Note that for UDDI registries, domain specific ontologies can be stored by the standard bodies who define them and the server, where the service is defined, can host the semantic description of the service instance.

The second key issue is how to relate the services advertised in the registry with the semantic defined through an ontology. The mechanism to relate semantics with services advertised in the UDDI registries are the tModel keys and the category bags of registry entries. tModels provide the ability to describe compliance with taxonomies, ontologies or controlled vocabularies. Therefore if tModel keys are assigned to the nodes of the ontology (for example given in Figure 3) and if the services put the corresponding tModel keys in their category bags, it is possible to locate services conforming to the semantic given in a particular node of this ontology. This issue is elaborated in [9].

An ebXML registry [13], on the other hand, allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "ClassificationScheme". Furthermore, "Classification" objects explicitly link the services advertised with the nodes of a "ClassificationScheme". This information can then be used to discover the services by exploiting the ebXML query mechanisms.

Consider for example the service Functionality Ontology given in Figure 3. Such a hierarchy can be stored in an ebXML registry through the piece of code as shown in Figure 8 Part (a), and then the registry objects can be related with the nodes in the hierarchy. In this way it is possible to give meaning to the services. In other words, by relating a service with a node in the 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, "Klinik_Bilgi_Saglayici". When we associate



```
<rim:ClassificationScheme id='HL7' isInternal='true' nodeType='uniqueCode'>
  <rim:Name> <rim:LocalizedString calue='WebService'/>. </rim:Name>
  <rim:Description>
   <rim:LocalizedString value='This is a sample HL7 WebServiceSchema'/>
  </rim:Description>
</rimClassificationScheme>

<rim:ClassificationNode id='PatientReferralServices' parent='HL7'>
  <rim:Name> <rim:LocalizedString calue='PatientReferralServices'/>   </rim:Name>
  <rim:Description/>
</rimClassificationNode>

<rim:ClassificationNode id='PatientInformationRequest' parent='PatientReferralServices'>
  <rim:Name> <rim:LocalizedString calue='PatientInformationRequest'/>   </rim:Name>
  <rim:Description/>
</rimClassificationNode>

<rim:ClassificationNode id='ClinicalInformation' parent='PatientInformationRequest'>
  <rim:Name> <rim:LocalizedString calue='ClinicalInformation'/>   </rim:Name>
  <rim:Description/>
</rimClassificationNode>

<rim:ClassificationNode id='GetClinicalInformation' parent='ClinicalInformationRequest'>
  <rim:Name> <rim:LocalizedString calue='GetClinicalInformation'/>   </rim:Name>
  <rim:Description/>
  <Slot name='PatientId' slotType='StringList'>
  <Slot name='ClinicalInformation' slotType='StringList'>
</rimClassificationNode>
```

```
<SubmitObjectsRequest>
  <rim:LeafRegistryObjectList>
     <Service id="Klinik_Bilgi_Saglayici">
        <Name> <LocalizedString lang="TR"
                   value="Klinik_Bilgi_Saglayici"/></Name>
        <Slot name='PatientID'>
                <valueList><value>PID</Value></ValueList>
        </Slot>
        <Slot name='ClinicalInformation'>
                <valueList><value>CI</Value></ValueList>
        </Slot>
        <ServiceBinding accessURI="http://.../Klinik_Bilgi_Saglayici_Servis>
        <SpesificationLink spesificationObject="wsdl" /> </ServiceBinding>
     </Service>
     <Classification classificationNode="GetClinicalInformation"
                ClassifiedObject="Klinik_Bilgi_Saglayici"/>
    <ExtrinsicObject id="wsdl" mimeType="text/xml" />
   </rim:LeafRegistryObject>
</SubmitObjectsRequest>
```

Fig. 8. Defining Ontology Classes in ebXML and Relating a Service Instance with the Ontology Class

"Klinik_Bilgi_Saglayici" with the "GetPatientClinicalInformation" node through a "SubmitObjectsRequest" as shown in Figure 8 Part (b), its meaning becomes clear; that this service is providing patient clinical information. Furthermore "Klinik_Bilgi_Saglayici" service inherits properties of the "GetPatientClinicalInformation" service such as "PatientID" and "ClinicalInformation".

Finally, how to store OWL ontologies into ebXML registries and how to associate these ontologies with Web services are described in [10].

3 System Architecture

The Artemis project addresses the interoperability problem in the healthcare domain where organisations have proprietary application systems to access data. To exchange information there are different standards like HL7, GEHR or CEN's ENV 13606. The aim of the Artemis project is to allow organizations keep their proprietary systems, yet expose the functionality through Web services. Furthermore, we propose an ontology based description of these data exchange standards. One of the goals of using ontologies is to reduce (or to eliminate) conceptual and terminological differences among the healthcare data exchange standards through semantic mediation.

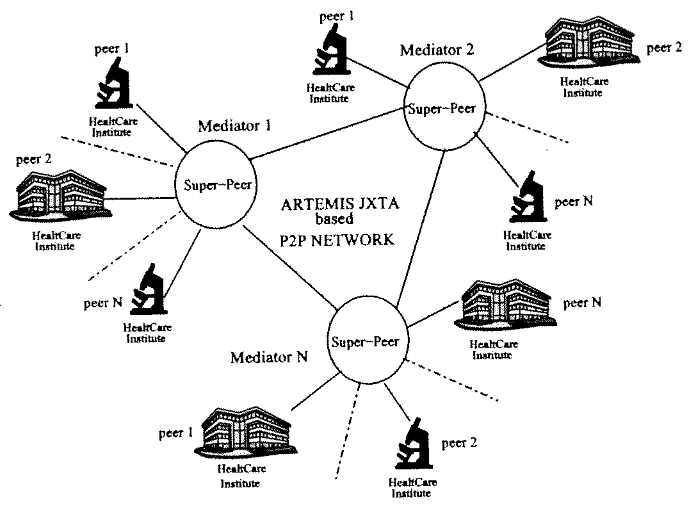


Fig. 9. Artemis P2P Architecture

Mediators are developed to process data from possibly several data sources and to prepare them for the effective use by applications [37]. However with WWW becoming the global communication medium and with the Semantic Web initiative, ontologies are becoming the primary part of the mediation process.

Artemis Web service architecture does not rely on globally agreed ontologies: rather healthcare institutes develop their own ontologies. However, it is reasonable to expect healthcare institutes to develop their own ontologies based on the concepts provided by the existing healthcare information standards since considerable semantic information is already captured there.

Artemis architecture then helps to reconcile the semantic differences among healthcare institutes through the mediator component. To provide scalability and discovery of other mediators, it has a P2P communication architecture. An overview of Artemis architecture is given in Figure 9.

3.1 Artemis Mediator P2P Architecture

In Artemis, healthcare institutes communicate with each other through mediators which resolve their differences bilaterally. When it comes to how to organize the mediators we make the following observations:

- The mediators must have a distributed architecture to provide for scalability.
- When a healthcare institute, say A, wants to communicate with another healthcare institute, say B, it should be possible to automatically locate the mediator of B.
- There are efficiencies to be gained by logically grouping the healthcare institutes which communicate often through a single mediator.

With these considerations in mind, Artemis mediators are organized as JXTA super peer groups. JXTA is an Open Source project [22] supported and managed by Sun Microsystems. Basically, JXTA is a set of XML based protocols to implement typical P2P functionalities. In the JXTA super peer based architecture, peers in a peer group communicate with their super peer to advertise their capabilities as well as to search for other peers.

In Artemis, each mediator is a super peer serving the healthcare institutes in its logical peer group. Super-peers employ keyword based routing indices where keywords are used to locate the healthcare institutes. On registration the peer provides this information to its super-peer.

3.2 Artemis Mediator Component

Generally speaking, semantic mapping is the process where two ontologies are semantically related at conceptual level and source ontology instances are transformed into target ontology entities according to those semantic relations. In Artemis, the source and target ontologies belong to the two healthcare institutes willing to exchange information. However, the mapping of these two ontologies are achieved through the reference ontologies stored in the mediator: the generic *Service Functionality* and *Service Message* ontologies. The mediator resolves the semantic differences between source and target ontologies by using these ontologies.

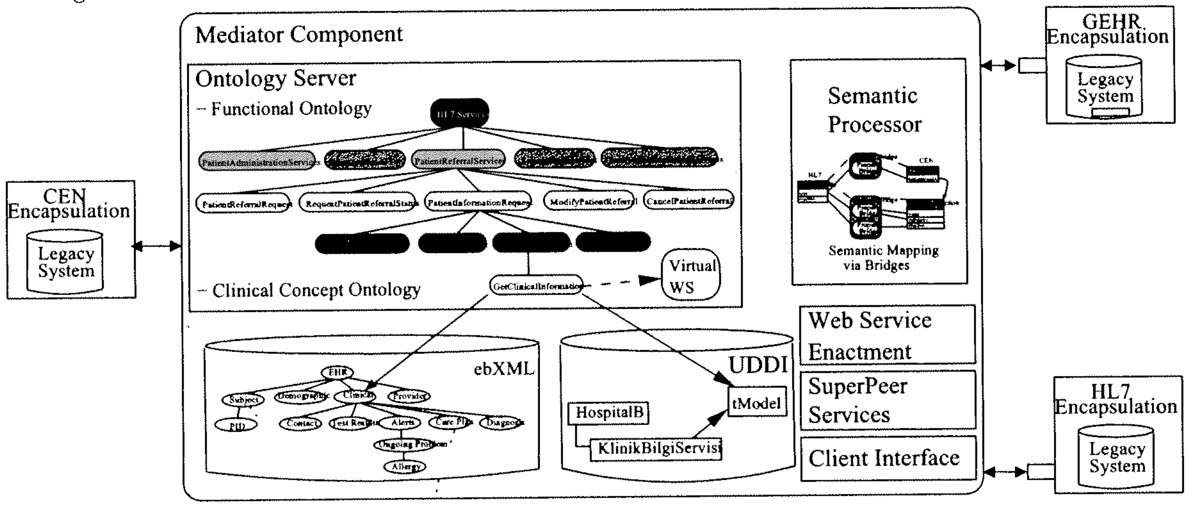


Fig. 10. An Overview of the Mediator

It should be noted that since all the ontologies involved are somehow related with the basic healthcare standards, the mediation process is simpler and hence more efficient. Furthermore, resolved semantic differences are stored as *Virtual Web Services (VWS)* to be reused as explained in Section 2.6.

The mediator architecture, which is shown in Figure 10, has the following subcomponents:

- Ontology server: The Ontology server contains the following ontologies:
  - *Service Functionality* and *Service Message* ontologies: Each healthcare institute may develop its own Service Functionality and Service Message ontologies based on existing healthcare information standards. The minimum requirement is annotating their services through such ontologies.
  - *Virtual Web Services* subsystem handles the creation of Virtual Web Services (VWSs) to provide complex aggregations of Web services. The creation of VWSs is realised according to the mappings between the ontologies of Web services' input and output semantics. Newly created VWSs are classified according to the *Service Functionality*

Ontology of the requesting party for its possible future reuse.

- *Semantic Processor*: There may be more than one *Service Functionality* and *Service Message* ontologies in the mediator and the mediator generates the mappings between them using its own reference ontologies based on the healthcare standards. In Artemis, MAFRA is used to represent the mappings and to transform the ontology instances. MAFRA uses the Semantic Bridge Ontology to define the mappings and includes a transformation engine. The mediator stores the previously defined mappings via semantic bridges. For example, the semantic equality relation between the "DiagnosticTestResult" concept in ENV 13606, and the "ObservationResult" concept in HL7 can be represented using MAFRA semantic bridges as follows:

```
<a:ConceptBridge rdf:ID="CB163312">
 <a:relatesTargetEntity rdf:resource=
  "http://www.srdc.metu.edu.tr/HL7#ObservationResult"/>
 <a:relatesSourceEntity rdf:resource=
  "http://www.srdc.metu.edu.tr/CEN#DiagnosticTestResult"/>
 <a:abstract rdf:resource="&a;True"/>
</a:ConceptBridge>
```

  Note that, more complex mappings can be represented using "semantic bridges", such as compositions, alternatives, and transformations aided by external functions.

  At runtime the source ontology instances are tranformed into target ontology instances by providing the source instance and the rdf representation of mapping to the transformation engine of MAFRA.

- Service registries like UDDI and ebXML: The Web services of the involved healthcare institutes are published in the UDDI or ebXML registries of the mediator.

- Web service Enactment Component handles the invocation of the Web Services and transmits the results of the Web Services. Bridge [4] is used to deploy and invoke Web services in JXTA environment.

- Superpeer Services Component contains the services that provides the communication with other Mediators in a P2P infrastructure. Basically, these services implement the JXTA Protocols. For example, Discovery Service that implements the JXTA Peer Discovery Protocol is used to find the other Mediators through a keyword based search mechanism.

- Client Interface handles the communication of healthcare institutes with the mediator using client-mediator protocol.

4 Related Work

Currently, describing the semantics of Web services is a very active research area. DAML-S [7] (later OWL-S) is a comprehensive effort defining an upper ontology for Web services. Service discovery through DAML-based languages is also addressed in the literature [8,24,25,28] where artificial intelligence techniques are used to discover services.

In [27], an RDF mapping meta-ontology, called RDF Translation (RDFT), is proposed which specifies a language for mapping XML DTDs to and from RDF Schemas for business integration tasks.

In ChattyWeb [1], the emerging P2P paradigm is seen as an opportunity to improve semantic interoperability, in particular in revealing new possibilities on how semantic agreements can be achieved. It is argued that establishing local agreements is a less challenging task than establishing global agreements by means of globally agreed schemas or shared ontologies. Once such local agreements exist, through the "semantic gossiping" process proposed, global agreements can be achieved in a P2P manner.

The work described in this paper has benefited from the previous work in the following areas:

- *Semantic Web Service Architecture:* The semantic architecture of Artemis is adapted from Semantic Web Enabled Web Services (SWWS) [5] architecture. In [5], the authors describe how semantics can be exploited in different levels of the Web service stack and stress the importance of ontologies and semantic mediation to deal with the interoperability problem. A detailed overview of the Web Service Modeling Framework (WSMF) is given in [14].

- *Ontology Mapping*: The ontology mapping component of Artemis mediator uses the technologies described in [16] where a semantic mapping and reconciliation engine is developed within the scope of the Harmonise project [15]. The Harmonise project aims to develop a harmonization network for tourism industry to allow participating tourism organisations to keep their proprietary data format and use ontology mediation while exchanging information in a seamless manner. For this purpose they have defined a *Interoperability Minimum Harmonization Ontology* and an interchange format for tourism industry. MAFRA [23] tool is used for ontology mediation.
- Extending the UDDI registries with semantic capabilities is addressed in [9], where we describe a mechanism to relate DAML-S ontologies with services advertised in the UDDI registries. [29] also addresses importing semantic to UDDI registries where DAML-S specific attributes such as *inputs, outputs* and *geographicRadius* are represented using tModel mechanisms of UDDI.

  In [10], how ebXML registries can be enriched with Web service semantic is described.
- Finally, some of the initial ideas on deploying Web services in the healthcare domain is presented in [11].

5 Conclusions and Future Work

Artemis is an EU funded project (IST-2103) which has been set up to develop and deploy semantically enriched services in the healthcare domain to provide interoperability. Through Artemis by introducing Web services to the healthcare domain, the access to electronic health records are standardized rather than standardizing the documents themselves.

We use the domain knowledge exposed by the existing healthcare informatics standards to define *Service Functionality* and *Service Message* ontologies. A *Service Functionality* ontology is used to specify the operational meanings of Web services and it is based on HL7. A *Service Message* ontology is used in specifying the semantics of Web service messages and is developed through electronic healthcare record based standards such as ENV 13606 and GEHR.

In Artemis, the healthcare institutes define their own ontologies based on the existing healthcare information standards. The mediator component of the system uses the *Service Functionality* and *Service Message* ontologies as references to resolve the semantic differences between two healthcare institutes. To provide for scalability and automated discovery of other mediators, the mediator architecture is based on a P2P framework, namely JXTA.

In this paper, we mainly focused on the clinical concept part of the message ontologies. Our main motivation for concentrating on clinical concept ontologies is that the electronic healthcare record based standards present detailed semantics in this regard. However healthcare is a many-to-many business. It is not only connecting a hospital to its branch clinics but to an array of internal and external agencies such as insurance entities, financial institutes and government agencies. Therefore there are other aspects of healthcare informatics such as billing and insurance that need to be covered. Our future work includes extending message ontologies with semantic concepts to handle these aspects including financial information.

References

[1] Aberer, K., Cudre-Mauroux, P., Hauswirth, M., "The Chatty Web: Emergent Semantics Through Gossiping", The Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, May 2003

[2] Artemis Project, http://www.srdc.metu.edu.tr/webpage/projects/artemis

[3] Beale, T., GEHR Object Model Architecture (GOM), http://www.gehr.org/technical/

model_architecture/model_architecture_4.1E.pdf

[4] Burton, Kevin A., Bridge, http://soap.jxta.org/servlets/ProjectHome

[5] Bussler, C., Fensel, D., Maedche, A., "A Conceptual Architecture for Semantic Web Enabled Web Services", SIGMOD Record, Vol. 31, No. 4, December 2002.

[6] CEN TC/251 (European Standardization of Health Informatics) ENV 13606, Electronic Health Record Communication, http://www.centc251.org/

[7] DAML-S, DAML Services Coalition, in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.

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

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

[10] Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004.

[11] Dogac, A., Laleci, G., Kirbas, S., Kabak, Y., Sinir, S., Yildiz, A., "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain", Extended Abstract, submitted for publication.

[12] R. H. Dolin, L. Alschuler, S. Boyer, and C.Beebe, "An Update on HL7's XML-based Document Representation Standards", http://www.amia.org/pubs/symposia/D200113.PDF

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

[14] Fensel, D., Bussler, C., "The Web Service Modeling Framework WSMF", Electronic Commerce Research and Applications, Vol. 1, Issue 2, Elsevier Science B.V., Summer 2002.

[15] Harmonise Project, IST-2000-29329, Tourism Harmonisation Network, http://www.harmonise.org/

[16] Harmonise Project, Deliverable 3.2: Semantic mapping and Reconciliation Engine subsystems, http://sourceforge.net/projects/hmafra

[17] The Good Electronic Health Record, http://www.gehr.org

[18] Health Level 7 (HL7), http://www.hl7.org

[19] HL7, Chapter 11 Patient Referral, http://www.hl7.org/library/General/v231.zip

[20] ICD-10, International Statistical Classification of Diseases and Related Health Problems, http://www.who.int/whosis/icd10/

[21] ISO TC/215, International Organization for Standardization, Health Informatics Technical Committee, http://www.iso.ch/iso/en/stdsdevelopmenttc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=4720

[22] Project JXTA, http://www.jxta.org/

[23] Maedche, A., Motik, D., Silva, N., Volz, R., "MAFRA-A MApping FRAmework for Distributed Ontologies", In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management EKAW-2002, Madrid, Spain, 2002.

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

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

[26] Motta, E., Domingue, J., Cabral, L., Gaspari, M., "IRS II: A Framework and Infrastructure for Semantic Web Services", 2nd International Semantic Web Conference, Florida, USA, October 2003.

[27] Omelayenko, B., Fensel, D., Bussler, C., "Mapping Technology for Enterprise Integration", Proc. of Special Track on Semantic Web at The 15th International FLAIRS

Conference, USA, May 2002.

[28] Paolucci, M., Kawamura, T., Payne, T., Sycara, K., "Semantic Matching of Web Services Capabilities", in Proc. of Intl. Semantic Web Conference, Sardinia, Italy, June 2002.

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

[30] Rahm, E., Bernstein, P.A, "A survey of approaches to automatic schema matching", VLDB J. Vol. 10, No. 4, Dec. 2001.

[31] Silva, N., Rocha, J., "Semantic Web Complex Ontology Mapping", Proceedings of the IEEE Web Intelligence 2003 conference, Canada, October 2003.

[32] SNOMED Clinical Terms, http://www.snomed.org/snomedct_txt.html

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

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

[35] Web Ontology Language OWL, http://www.w3.org/TR/owl-features/

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

[37] Wiederhold, G., "Mediators in the Architecture of Future Information Systems", IEEE Computer, Vol.25 No.3, March 1992.

Providing Semantic Interoperability in the Healthcare Domain through Ontology Mapping[*]

Veli Bicer, Gokce Banu Laleci, Asuman Dogac, Yildiray Kabak
Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
email: *asuman@srdc.metu.edu.tr*

Abstract: One of the most prominent European strategic objectives in eHealth is to provide interoperability among healthcare information systems. In this paper, we describe an engineering approach to semantic interoperability to provide the exchange of meaningful clinical information among healthcare institutes. The approach is generic enough to be used between any medical information systems but we demonstrate the inter workings of the developed prototype by mediating between the two incompatible versions of HL7, namely, Version 2 and Version 3.

We address the interoperability problem by first defining the HL7 Version 2 and Version 3 message ontologies in OWL and mapping them one another using the OWL mapping tool developed, called OWLmt. Given an ontology mapping between HL7 Version 2 and Version 3, OWLmt automatically transforms the instances of the messages exchanged. We note that in a realistic healthcare setting today, the exchanged message instances are EDI or XML, not messages conforming to an ontology. Therefore additional tools are incorporated into the system for converting EDI messages to XML messages, generating XML Schemas from XML documents, and converting XML schemas and messages into OWL.

1. Introduction

CEN/ISSS eHealth Standardisation Focus Group has identified the most prominent strategic aims of healthcare informatics in Europe as follows [9]:

- Improving access to clinical records;
- Enabling patient mobility and cross border access to healthcare;
- Reducing clinical errors and improving safety;
- Improving access to quality information on health for patients and professionals;
- Improving efficiency of healthcare processes.

All of these objectives require the interoperability of healthcare information systems whereas most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. A number of standardization efforts are progressing to address this problem such as EHRcom [3], openEHR [14] and HL7 Version 3 [8]. Yet, since it is not realistic to expect all the healthcare institutes to conform to a single standard, there is a need to address the interoperability at the semantic level. Semantic interoperability is the ability for information shared by systems to be understood

[*] This work is supported by the European Commission through IST-1-002103-STP Artemis Project and in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 104E013

at the level of formally defined domain concepts so that the information is computer processable by the receiving system [10].

In this paper, we describe an engineering effort developed within the scope of the Artemis project [1] to provide the exchange of meaningful clinical information among healthcare institutes. For this purpose, the existing applications are wrapped as Web services. Then, by use of an OWL ontology mapping tool, called OWLmt, the messages are semantically mediated to provide interoperability.

This approach is generic enough to provide interoperability between any information systems. However the prototype developed to demonstrate its feasibility, currently mediates between HL7 Version 2 and Version 3 messages since it uses some application specific tools such as HL7 HAPI (HL7 application programming interface) to generate OWL [15] message instances from the EDI messages.

2. System Architecture

HL7 version 2 is the most widely implemented healthcare informatics standard in the world today. Yet being HL7 Version 2 compliant does not imply direct interoperability between healthcare systems. Version 2 messages contain many optional data fields. This optionality provides great flexibility, but necessitates detailed bilateral agreements among the healthcare systems to achieve interoperability. To remedy this problem, HL7 [6] has developed Version 3 which is based on an object-oriented data model, called Reference Information Model (RIM) [7]. However HL7 Version 3 messages are not interoperable with HL7 Version 2 messages. Hence, the major challenge has become the interoperability of HL7 Version 3 with the Version 2.x implementations.

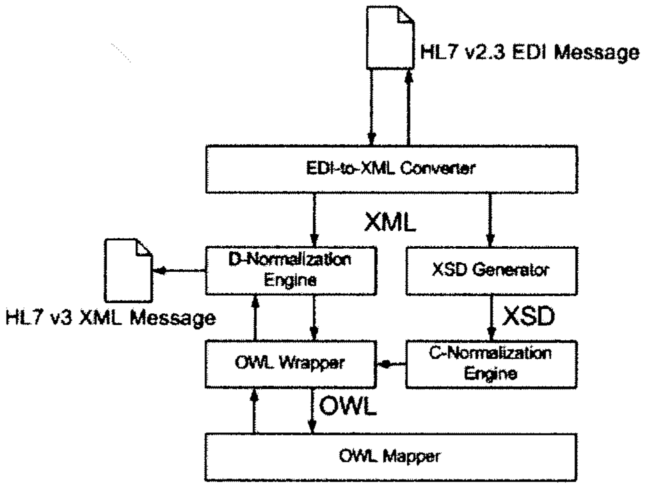


Figure 1. The Overall System Architecture

In this paper, we address this problem and show how HL7 Version 3 messages can be semantically mediated with HL7 Version 2 messages. The overall system architecture, as shown in Figure 1, involves the following components:

- *OWL Ontology Mapping Tool (OWLmt)*: The main component of the architecture is the OWL mapping tool. The mappings created through this tool are used for transforming the instances of the source ontology into target ontology instances. Given that in a realistic healthcare setting today, the exchanged message instances EDI or XML, not messages conforming to an ontology, the source messages in EDI format need to be converted into OWL message instances and the target ontology message instances need to be converted to XML. Furthermore, there is a need for automatic generation of OWL Schemas from XML Schema Definitions (XSDs). The rest of the components in the system are used for these purposes.

- *EDI to XML Converter*: Since HL7 version 2 mostly uses EDI messages, these messages need to be converted to XML first. The open-source programming library from HL7, namely, HAPI (HL7 application programming interface) is used in transforming the EDI messages into their XML representations.
- *XML Schema Generator*: For generating the XML Schemas of the resultant messages, Castor's XMLInstance2Schema tool [2] is used.
- *C-Normalization engine*: Conceptual Normalization (C-Normalization) engine of the Harmonise project [5] is used to parse the XML Schema, and create the corresponding RDFS schema [17].
- *OWL Wrapper*: Since OWLmt uses OWL Schemas instead of RDFS schemas, an OWL wrapper is developed using Jena API [11] to create OWL schemas out of RDFS files after the C-Normalization step.
- *D-Normalization engine*: Data Normalization (D-Normalisation) Engine transforms the data instances from XML to OWL or OWL to XML. In this step, the output of the C-Normalization step, "Normalization Map", is given as an input to describe how each component in XSD can be transformed into a component in RDFS and vice-versa.

  In the following sections, all of these components are described in detail.

2.1. OWL Mapping Tool: OWLmt

Ontology Mapping is the process where two ontologies with an overlapping content are related at the conceptual level, and the source ontology instances are automatically transformed into the target ontology instances according to these relations. We have developed an OWL mapping tool, called OWLmt, to handle ontology mediation by mapping the OWL ontologies in different structure but with an overlapping content one into other. The architecture of the system, as shown in Figure 2, allows mapping patterns to be specified through a GUI tool based on a Mapping Schema. The Mapping Schema, as shown in Figure 3, is also defined in OWL. The mapping engine uses the mapping patterns specified through the GUI to automatically transform source ontology instances into target ontology instances.

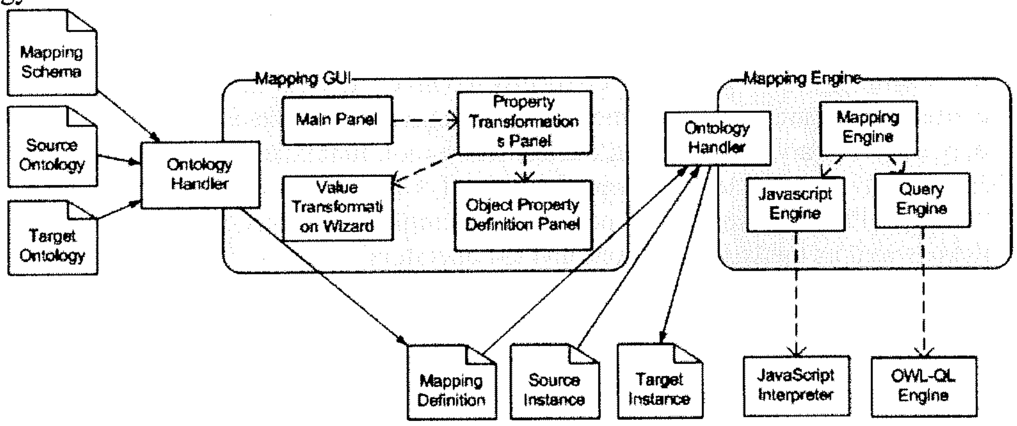


Figure 2. The Architecture of OWLmt

Mapping patterns basically involve the following:

- *Matching the source ontology classes to target ontology classes*: In order to represent the matching between the classes of source and target ontologies, we have defined four mapping patterns: *EquivalentTo, SimilarTo, IntersectionOf* and *UnionOf.* Two identical classes are mapped through E*quivalentTo* pattern. *SimilarTo* pattern implies that the involved classes have overlapping content. How these two classes are related is determined through further mapping of their datatype properties and object properties.

The *IntersectionOf* pattern creates the corresponding instances of the target class as the intersection of the declared class instances. Similarly, the *UnionOf* pattern takes the union of the source classes' instances to create the corresponding instances of the target class.
Furthermore, a class in a source ontology can be a more general (super class) of a class in the target ontology. In this case, which instances of the source ontology makes up the instances of the target ontology is defined through KIF conditions to be executed by the mapping engine. When a source ontology class is a more specific (sub class) of a target ontology class, all the instances of the source ontology qualify as the instances of the target ontology.

- *Matching the source ontology Object Properties to target ontology Object Properties*: In addition to matching a single object property in the source ontology with a single object property in the target ontology, in some cases, more than one object properties in the source ontology can be matched with one or more object properties in the target ontology. Therefore, OWLmt allows defining "ObjectPropertyTransform" pattern which represents the path of classes connected with object properties. Paths are defined as triples in KIF [13] format and executed through the OWL-QL [16] engine. Through such patterns, the OWLmt constructs the specified paths among the instances of the target ontology in the execution step based on the paths defined among the instances of the source ontology.
- *Matching source ontology Data Properties to target ontology Data Properties*: Specifying the "DatatypePropertyTransform" helps to transform datatype properties of an instance in the source ontology to corresponding target ontology instance datatype properties. Since the datatype properties may be structurally different in source and target ontologies, more complex transformation operations may be necessary than copying the data in source instance to the target instance. XPath specification [18] defines a set of basic operators and functions which are used by the OWLmt such as "concat", "split", "substring", "abs", and "floor". In some cases, there is a further need for a programmatic approach to specify complex functions. For example, the use of conditional branches (e.g. if-then-else, switch-case) or iterations (e.g while, for-next) may be necessary in specifying the transformation functions. Therefore, we have added JavaScript support to OWLmt. By specifying the JavaScript to be used in the "DatatypePropertyTransform" pattern, the complex functions can also be applied to the data as well as the basic functions and the operators.

2.1.1 OWLmt Mapping Schema

The mapping patterns used in the OWLmt are defined through an OWL ontology called "Mapping Schema". Each mapping pattern is an owl:class in the "Mapping Schema" as shown in Figure 3. The additional information needed in the execution of the patterns such as "inputPath" and "outputPath" for ObjectProperty Transform pattern are defined as properties of this class. The "inputPath" and "outputPath" datatype properties hold the query strings in the KIF format which are used in the execution to query the source ontology instances in order to build the target instances.

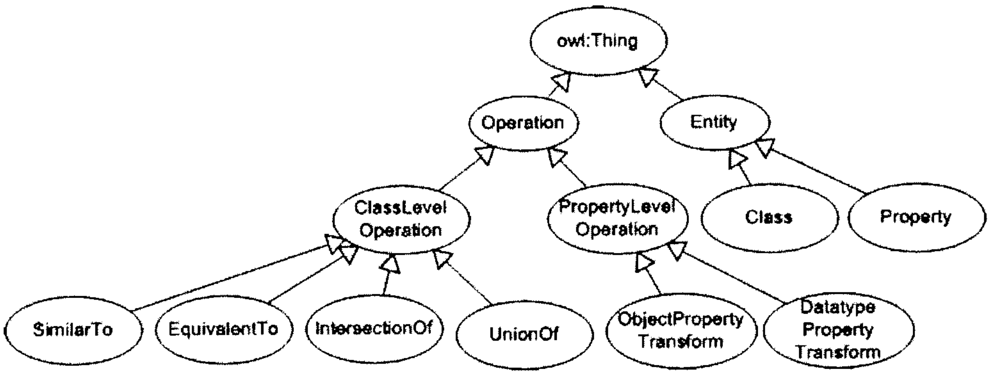


Figure 3. OWL Mapping Schema

2.1.2 OWLmt GUI

OWLmt GUI, as shown in Figure 4, allows the user to define the mapping patterns. It consists of five components: Ontology Handler, Main Panel, Property Transformations Panel, Value Transformation Wizard and Object Property Definition Panel. The Ontology Handler is used in parsing and serializing the ontology documents. The class mapping patterns are defined in the main panel. The property mapping patterns are defined in the property transformation panel. This panel lets the user to create new property mapping patterns such as the "ObjectPropertyTransform" and "DatatypePropertyTransform". The value transformation wizard is used to configure a "DatatypePropertyTransform" pattern. By using this wizard, the functions used in the value transformation of the datatype properties can be specified.

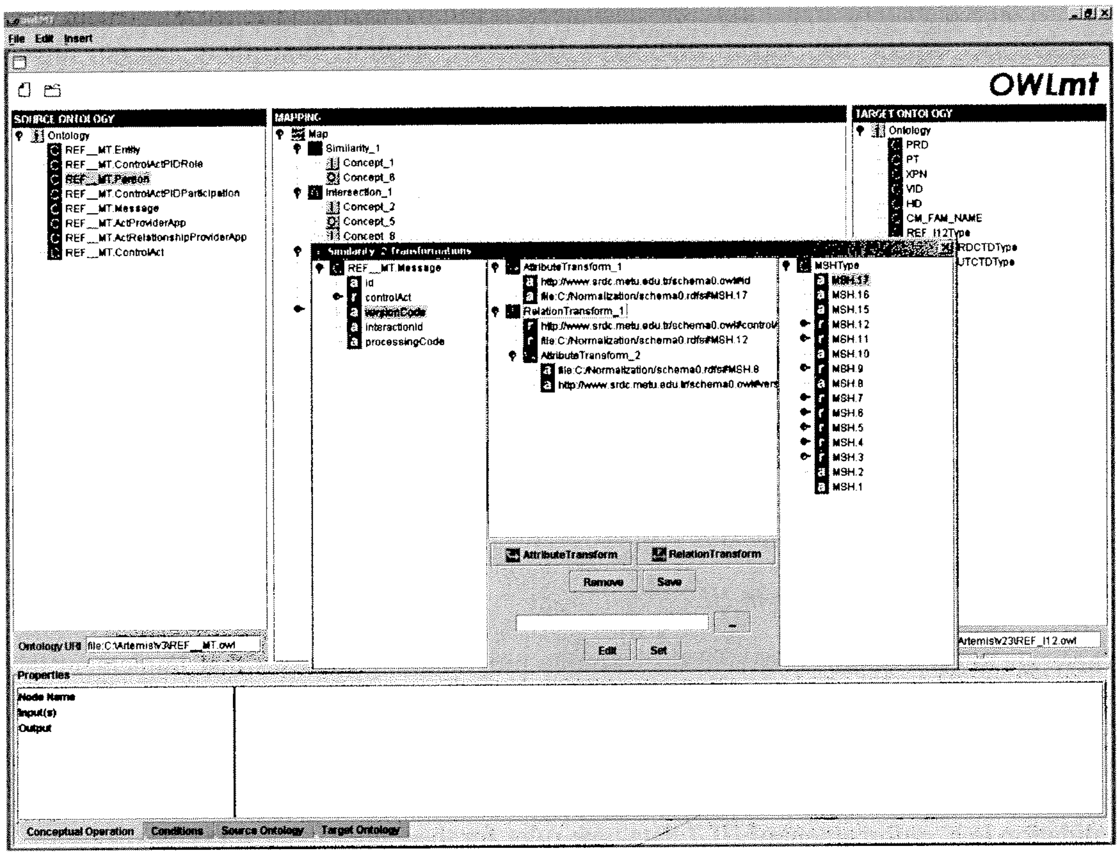


Figure 4. OWLmt GUI

2.1.3 OWLmt Engine

The mapping engine is responsible for creating the target ontology instances using the mapping patterns and the instances of the source ontology. It uses OWL Query Language (OWL-QL) to retrieve required data from the source ontology instances. OWL-QL is a joint US/EU initiative to develop a query language for OWL [16]. While executing the class and property mapping patterns, the query strings defined through the mapping GUI are send to the OWL-QL engine with the URL of the source ontology instances. The query engine executes the query strings and returns the query results.

The OWL-QL engine uses the JTP reasoning engine [12], an object-oriented modular reasoning system. The system consists of the modules called reasoners classified into "asking reasoners" and "telling reasoners" according to their functionality. The "asking reasoners" process queries and return proofs for the answers while the "telling reasoners" process assertions and proofs and draw conclusions. The modularity of the system enables it to be extended by adding new reasoners or customizing existing ones.

The use of the OWL-QL enables OWLmt to have reasoning capabilities. When querying the source ontology instances or while executing the KIF patterns, OWL-QL reasons over the explicitly stated facts to infer new information. As an example, consider two instances, I1 and I2, which are the members of the classes C1 and C2 respectively. If these two instances are related with the "owl:sameAs" construct, one of them should be in the extension of the intersection class, say C3, of the classes C1 and C2. Hence, the *IntersectionOf* pattern transforms the instance I1 and I2 to the instance I3 which is a member of C3 in the target ontology. However, assume that there is no direct "owl:sameAs" construct but there is a functional property which implies that these two instances are the same. The reasoning engine can infer from the definition of the "owl:FunctionalProperty" by using the rule;

- (rdf:type ?prop owl:FunctionalProperty) (?prop ?instance ?I1) (?prop ?instance ?I2) →(owl:sameAs ?I1 ?I2)

that the instances I1 and I2 are the same instance resulting in the instance I3 to be in the target ontology.

After executing the class mapping patterns, the mapping engine executes the property mapping patterns. Similar to the class mapping patterns, OWL-QL queries are used to locate the data. In order to perform value transformations, the mapping engine uses the JavaScripts in the "DatatypePropertyTransform" pattern. To execute the JavaScripts, an interpreter is used. The engine prepares the JavaScript by providing the values for the input parameters and sends it to the interpreter. The interpreter returns the result, which is then inserted as the value of the datatype property in the target ontology instance.

2.2. EDI to XML Conversion in HL7

There are several commercial and open-source programming libraries that implement the HL7 standards. In our architecture, HAPI [4] (HL7 Application Programming Interface) Assembler/Disassembler Tool is used to transform the EDI messages into their XML representations. HAPI provides open source libraries for parsing and manipulating both EDI and XML messages that are HL7 conformant. Furthermore the library enables message validation (e.g. enforcement of HL7 data type rules for the values in the messages).

2.3. Normalization Tool

As previously mentioned, currently the healthcare application messages are usually in XML or EDI format (which can be converted to XML). Hence there is a need for automatic bidirectional transformation of XML message instances to OWL message instances as well as automatic generation of OWL Schemas from XML Schema Definitions (XSDs). Such a transformation, called Normalization, has been realized within the scope of the Harmonise project [5].

The "Normalization Engine" of the Harmonise project is used in generating RDFS schemas from local XSD schemas. This step is called Conceptual Normalization (C-Normalization) phase where the C-Normalization engine parses the XML Schema, and using a set of predefined "Normalization Heuristics", creates the corresponding RDFS schema components for each XML Schema component automatically. Normalization Heuristics define how specific XML Schema construct (e.g. complex type definition) can be projected onto a RDFS construct (entity or set of related entities) [5]. This process produces a "Normalization Map" which defines the associations between the XML Schema and the re-engineered RDFS model.

The second step in the Normalization process is the Data Normalization Process (D-Normalization) which is used for transforming the data instances from XML to OWL or OWL to XML. In this step, the output of the C-Normalization step, "Normalization Map", is used to guide transforming each component in XSD to a component in RDFS or vice-versa.

In Artemis architecture, we have used the Harmonise Normalization Engine. However since we need OWL Schemas instead of RDFS schemas, we developed an OWL wrapper using Jena API to create OWL schemas from the RDFS files after the C-Normalization step. Additionally in the D-Normalization step, through the same wrapper, the generated RDF instances are further translated in to OWL instances or vice versa as depicted in Figure 5.

Note that in Harmonise C-Normalization step, the enumeration of property values or basic data types defined in XML Schemas cannot be preserved. To handle this, the OWL Wrapper developed carries the enumeration of property values and basic data types to the OWL Schema. The enumerated classes are represented using <owl:oneOf rdf:parseType="Collection"> construct in case of enumerated classes, and using <owl:oneOf> <rdf:List> construct in case of enumerated datatypes. The data types are respresented by referring to XML Schema datatypes using RDF datatyping scheme.

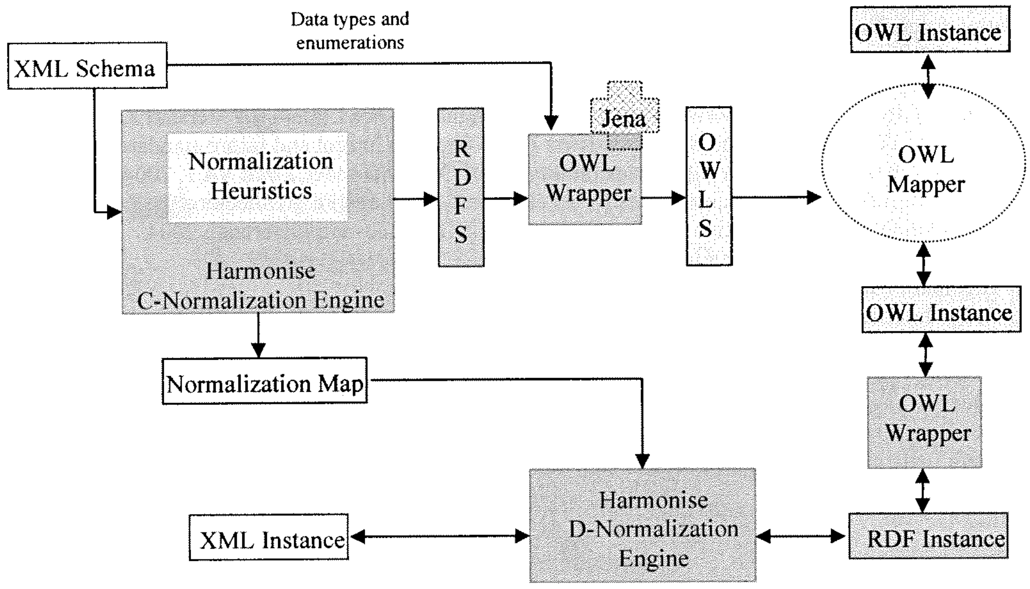


Figure 5. Normalization process for the bidirectional transformation of XML instances to OWL instances

3. Conclusions

As identified by CEN/ISSS eHealth Standardisation Focus Group [9], one of the most challenging problems in healthcare domain today is providing interoperability among healthcare information systems. In order to tackle this problem, we propose an engineering approach to semantic interoperability within the scope of the Artemis project. For this purpose, the existing applications are wrapped as Web services and the messages they exchange are then mediated through an ontology mapping tool developed, namely, OWLmt. One of the major contributions of the OWLmt is the use of OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the graphically defined mapping patterns.

Although the platform proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use in the healthcare domain, we have chosen to mediate between HL7 Version 2 and HL7 Version 3 messages to demonstrate the functionalities of the proposed platform. Since neither version 2, nor version 3 messages are ontology instances, their message structures, EDI and XML respectively, are normalized to OWL before OWL mapping process. Additional tools exploited for this purpose have also been elaborated in the paper.

References

[1] Artemis – A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information Systems, http://www.srdc.metu.edu.tr/webpage/projects/artemis/ .
[2] Castor's XMLInstance2Schema tool, http://castor.exolab.org/.
[3] ENV 13606:2000 "Electronic Healthcare Record Communication", http://www.centc251.org/TCMeet/doclist/TCdoc00/N00-048.pdf.
[4] HL7 Application Programming Interface, http://hl7api.sourceforge.net
[5] Harmonise, IST–2000-29329, Tourism Harmonisation Network, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.
[6] Health Level 7, http://www.hl7.org.
[7] HL7 Reference Information Model, http://www.hl7.org/library/data-model/RIM/modelpage_mem.htm.
[8] HL7 Version 3 Specification, http://www.hl7.org/library/standards_non1.htm#HL7 Version 3.
[9] Report from the CEN/ISSS eHealth Standardization Focus Group "Current and future standardization issues in the e-Health domain: Achieving interoperability", Part One: Main text, Draft V4.1, 2004-08-16.
[10] ISO/TS Health Informatics – Requirements for an electronic health record architecture, Technical Specification, International Organization for Standardization (ISO), Geneva, Switzerland, 2004.
[11] Jena Framework, http://jena.sourceforge.net/ .
[12] Java Theorem Prover, http://www.ksl.stanford.edu/software/JTP/ .
[13] Knowledge Interchange Format, http://logic.stanford.edu/kif/kif.html .
[14] OpenEHR Foundation, http://www.openehr.org/ .
[15] Web Ontology Language, http://www.w3.org/TR/owl-features/.
[16] OWL Query Language, http://ksl.stanford.edu/projects/owl-ql/ .
[17] Resource Description Framework Schema, http://www.w3.org/TR/rdf-schema/ .
[18] XML Path Language, http://www.w3.org/TR/xpath .

as automatic generation of OWL Schemas from XML Schema Definitions (XSDs). Such a transformation, called Normalization, has been realized within the scope of the Harmonise project [5].

The "Normalization Engine" of the Harmonise project is used in generating RDFS schemas from local XSD schemas. This step is called Conceptual Normalization (C-Normalization) phase where the C-Normalization engine parses the XML Schema, and using a set of predefined "Normalization Heuristics", creates the corresponding RDFS schema components for each XML Schema component automatically. Normalization Heuristics define how specific XML Schema construct (e.g. complex type definition) can be projected onto a RDFS construct (entity or set of related entities) [5]. This process produces a "Normalization Map" which defines the associations between the XML Schema and the re-engineered RDFS model.

The second step in the Normalization process is the Data Normalization Process (D-Normalization) which is used for transforming the data instances from XML to OWL or OWL to XML. In this step, the output of the C-Normalization step, "Normalization Map", is used to guide transforming each component in XSD to a component in RDFS or vice-versa.

In Artemis architecture, we have used the Harmonise Normalization Engine. However since we need OWL Schemas instead of RDFS schemas, we developed an OWL wrapper using Jena API to create OWL schemas from the RDFS files after the C-Normalization step. Additionally in the D-Normalization step, through the same wrapper, the generated RDF instances are further translated in to OWL instances or vice versa as depicted in Figure 5.

Note that in Harmonise C-Normalization step, the enumeration of property values or basic data types defined in XML Schemas cannot be preserved. To handle this, the OWL Wrapper developed carries the enumeration of property values and basic data types to the OWL Schema. The enumerated classes are represented using <owl:oneOf rdf:parseType="Collection"> construct in case of enumerated classes, and using <owl:oneOf> <rdf:List> construct in case of enumerated datatypes. The data types are respresented by referring to XML Schema datatypes using RDF datatyping scheme.

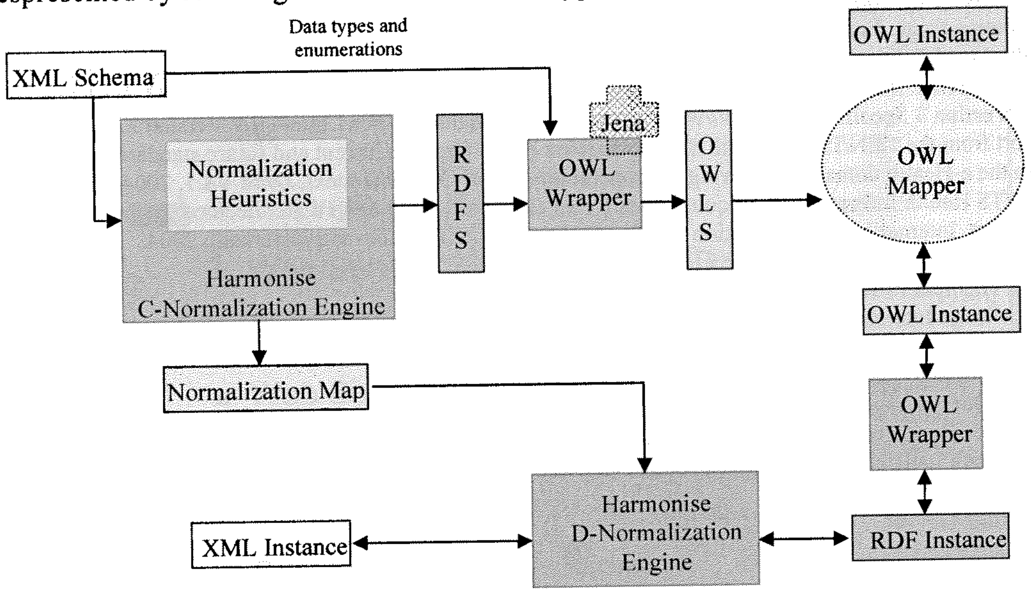


Figure 5. Normalization process for the bidirectional transformation of XML instances to OWL instances

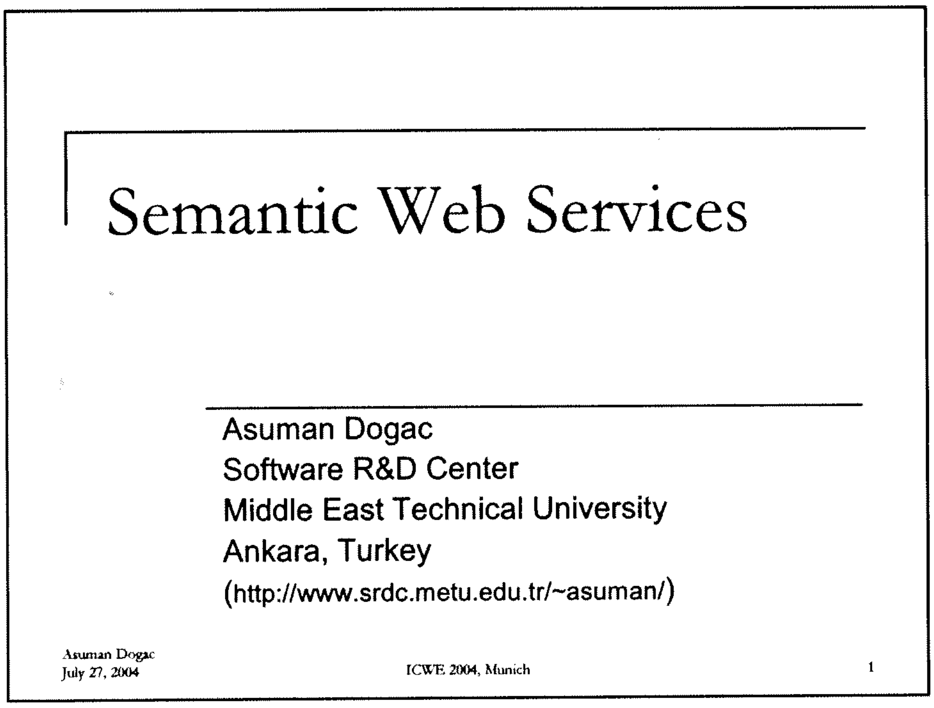
3. Conclusions

As identified by CEN/ISSS eHealth Standardisation Focus Group [9], one of the most challenging problems in healthcare domain today is providing interoperability among healthcare information systems. In order to tackle this problem, we propose an engineering approach to semantic interoperability within the scope of the Artemis project. For this purpose, the existing applications are wrapped as Web services and the messages they exchange are then mediated through an ontology mapping tool developed, namely, OWLmt. One of the major contributions of the OWLmt is the use of OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the graphically defined mapping patterns.

Although the platform proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use in the healthcare domain, we have chosen to mediate between HL7 Version 2 and HL7 Version 3 messages to demonstrate the functionalities of the proposed platform. Since neither version 2, nor version 3 messages are ontology instances, their message structures, EDI and XML respectively, are normalized to OWL before OWL mapping process. Additional tools exploited for this purpose have also been elaborated in the paper.

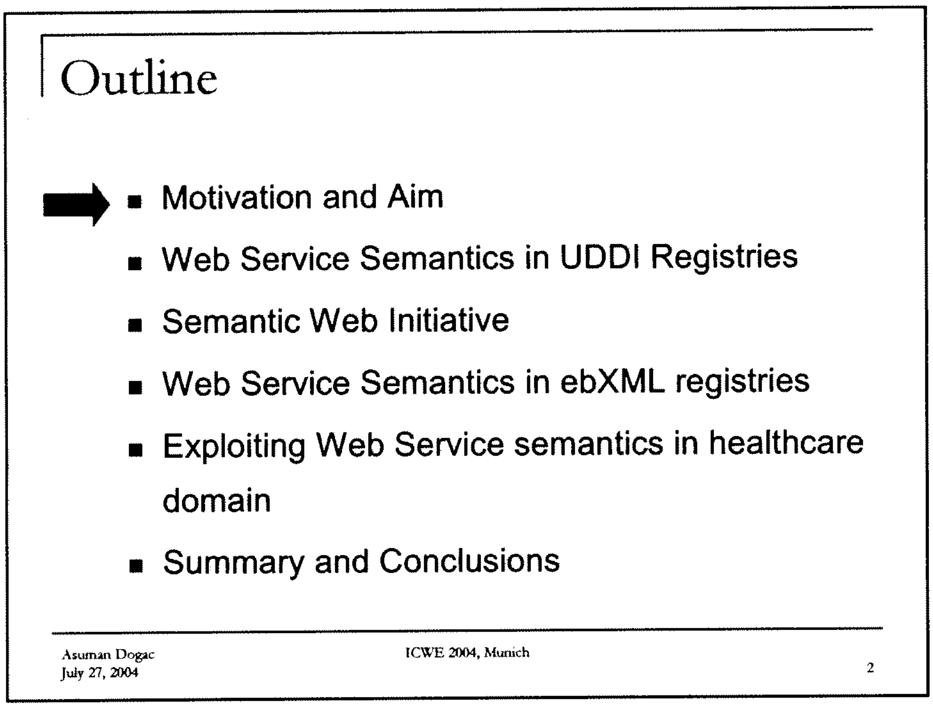
References

[1] Artemis – A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information Systems, http://www.srdc.metu.edu.tr/webpage/projects/artemis/ .
[2] Castor's XMLInstance2Schema tool, http://castor.exolab.org/.
[3] ENV 13606:2000 "Electronic Healthcare Record Communication", http://www.centc251.org/TCMeet/doclist/TCdoc00/N00-048.pdf.
[4] HL7 Application Programming Interface, http://hl7api.sourceforge.net
[5] Harmonise, IST–2000-29329, Tourism Harmonisation Network, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.
[6] Health Level 7, http://www.hl7.org.
[7] HL7 Reference Information Model, http://www.hl7.org/library/data-model/RIM/modelpage_mem.htm.
[8] HL7 Version 3 Specification, http://www.hl7.org/library/standards_non1.htm#HL7 Version 3.
[9] Report from the CEN/ISSS eHealth Standardization Focus Group "Current and future standardization issues in the e-Health domain: Achieving interoperability", Part One: Main text, Draft V4.1, 2004-08-16.
[10] ISO/TS Health Informatics – Requirements for an electronic health record architecture, Technical Specification, International Organization for Standardization (ISO), Geneva, Switzerland, 2004.
[11] Jena Framework, http://jena.sourceforge.net/ .
[12] Java Theorem Prover, http://www.ksl.stanford.edu/software/JTP/ .
[13] Knowledge Interchange Format, http://logic.stanford.edu/kif/kif.html .
[14] OpenEHR Foundation, http://www.openehr.org/ .
[15] Web Ontology Language, http://www.w3.org/TR/owl-features/.
[16] OWL Query Language, http://ksl.stanford.edu/projects/owl-ql/ .
[17] Resource Description Framework Schema, http://www.w3.org/TR/rdf-schema/ .
[18] XML Path Language, http://www.w3.org/TR/xpath .



Semantic Web Services

Asuman Dogac
Software R&D Center
Middle East Technical University
Ankara, Turkey
(http://www.srdc.metu.edu.tr/~asuman/)



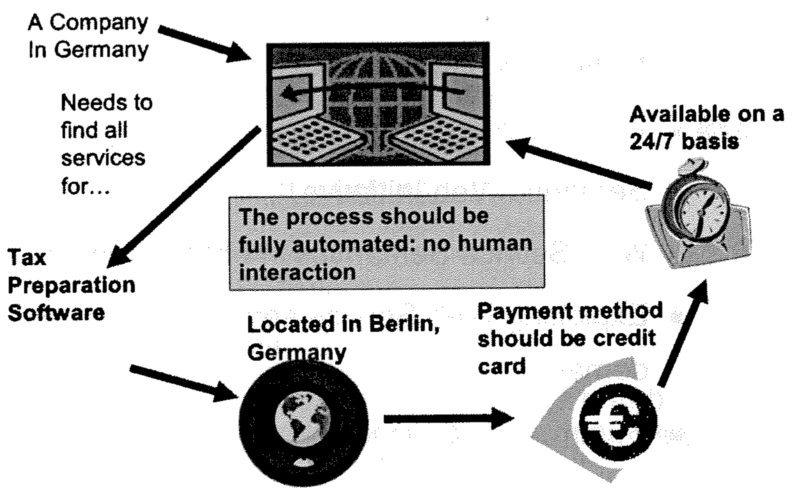
Outline

Motivation: Why do we need the semantics of Web services?

Why do we need Web Service Semantics?

- In order to exploit services in their full potential their properties must be defined:
  - **The methods of charging and payment**
  - **The channels by which the service is requested and provided**
  - **Constraints on temporal and spatial aspects**
  - **Availability**
  - **Service quality**
  - **Security, trust and rights attached to a service**
  - **And many more...**
- **Ref: O'Sullivan, J., Edmond, D., Hofstede, A., "What's in a Service? Towards Accurate Description of Non-Functional Service Properties", in the Journal of Distributed and Parallel Databases, Vol. 12, No. 2/3, Sept./Nov. 2002**

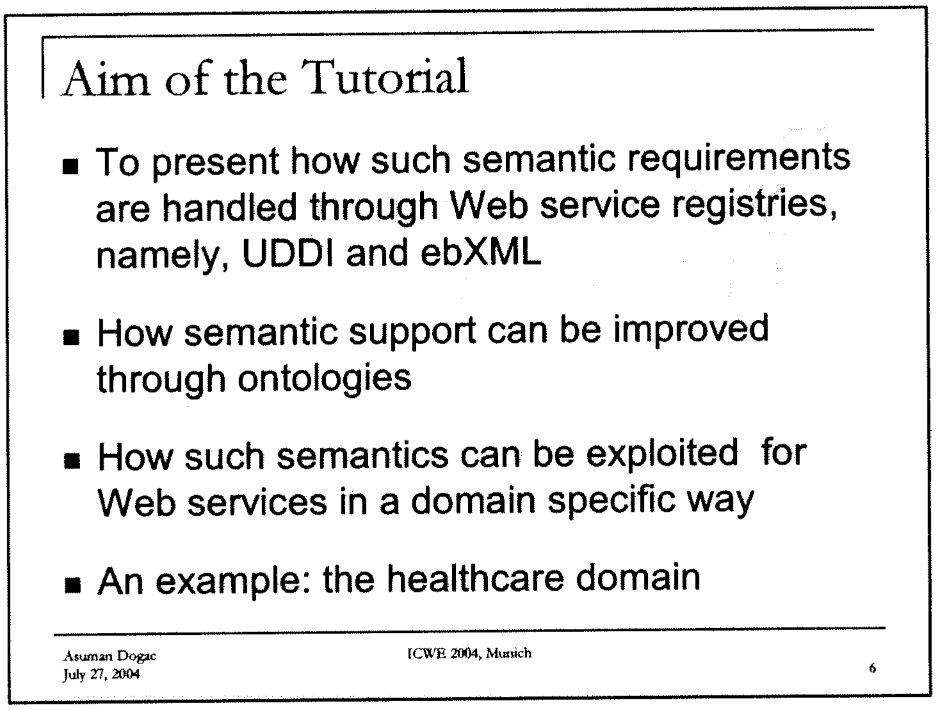
A Motivating Example


A Company In Germany
Needs to find all services for...
Tax Preparation Software
The process should be fully automated: no human interaction
Available on a 24/7 basis
Located in Berlin, Germany
Payment method should be credit card

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

5



Aim of the Tutorial

- To present how such semantic requirements are handled through Web service registries, namely, UDDI and ebXML
- How semantic support can be improved through ontologies
- How such semantics can be exploited for Web services in a domain specific way
- An example: the healthcare domain

Asuman Dogac
July 27, 2004

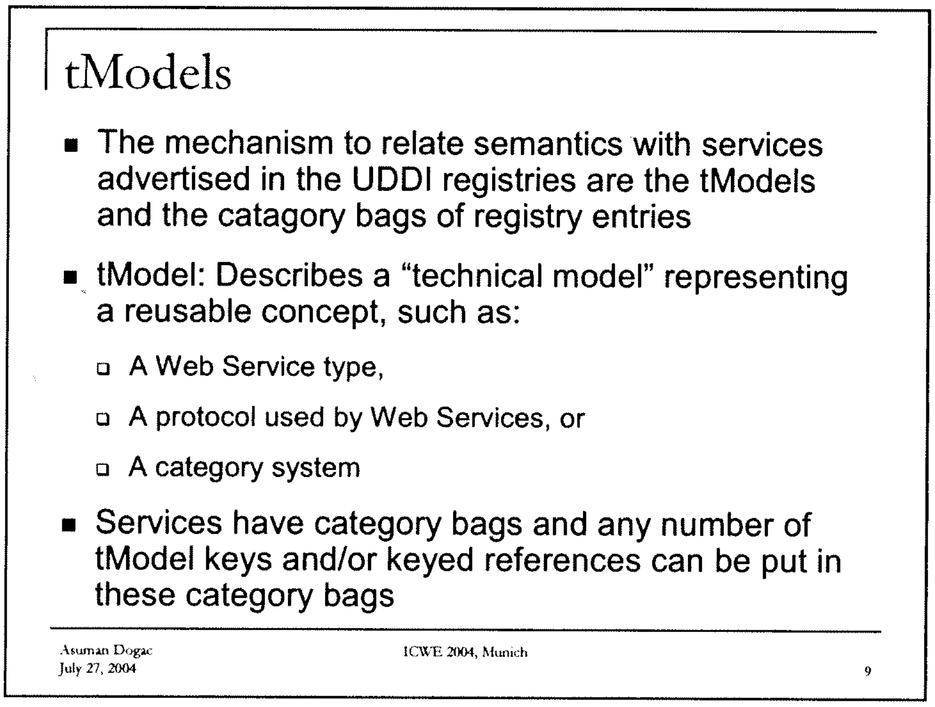
ICWE 2004, Munich

6

Outline

- Motivation and Aim
- ➡ Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions

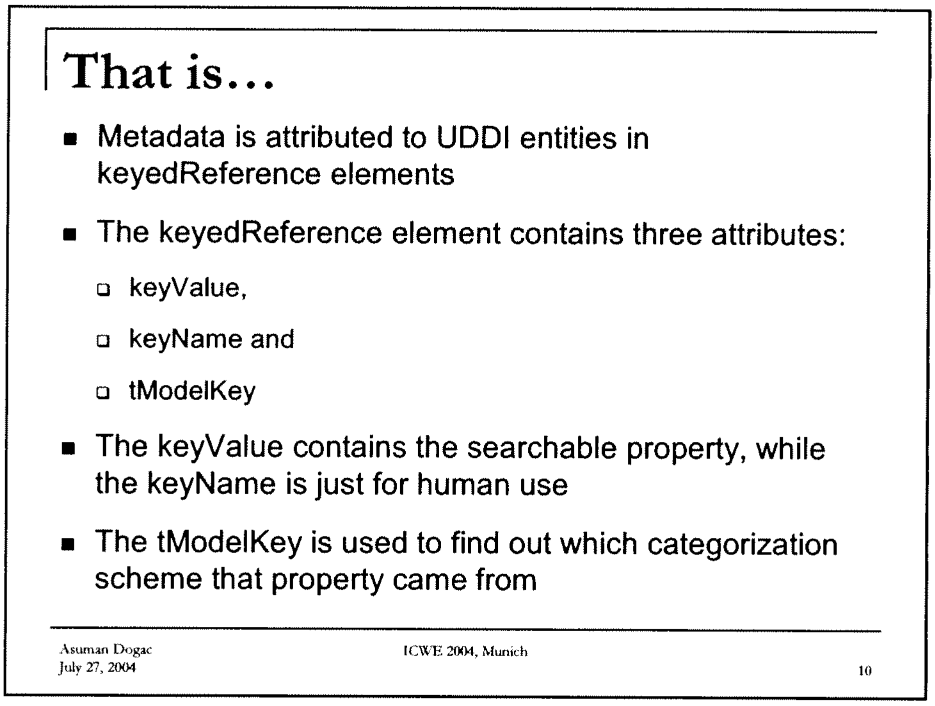
Web Service Semantics in UDDI Registries



tModels

- The mechanism to relate semantics with services advertised in the UDDI registries are the tModels and the catagory bags of registry entries
- tModel: Describes a “technical model” representing a reusable concept, such as:
  - A Web Service type,
  - A protocol used by Web Services, or
  - A category system
- Services have category bags and any number of tModel keys and/or keyed references can be put in these category bags

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
9



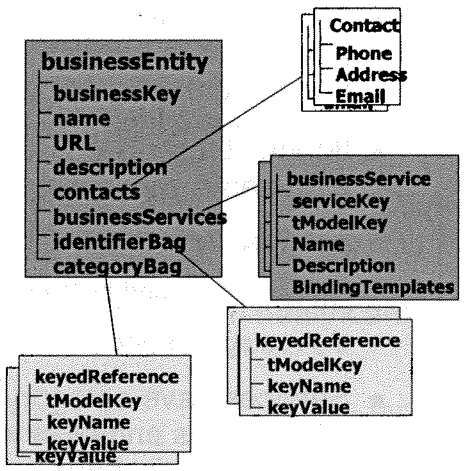
That is...

- Metadata is attributed to UDDI entities in keyedReference elements
- The keyedReference element contains three attributes:
  - keyValue,
  - keyName and
  - tModelKey
- The keyValue contains the searchable property, while the keyName is just for human use
- The tModelKey is used to find out which categorization scheme that property came from

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
10

Defining Service Semantics in UDDI Registries

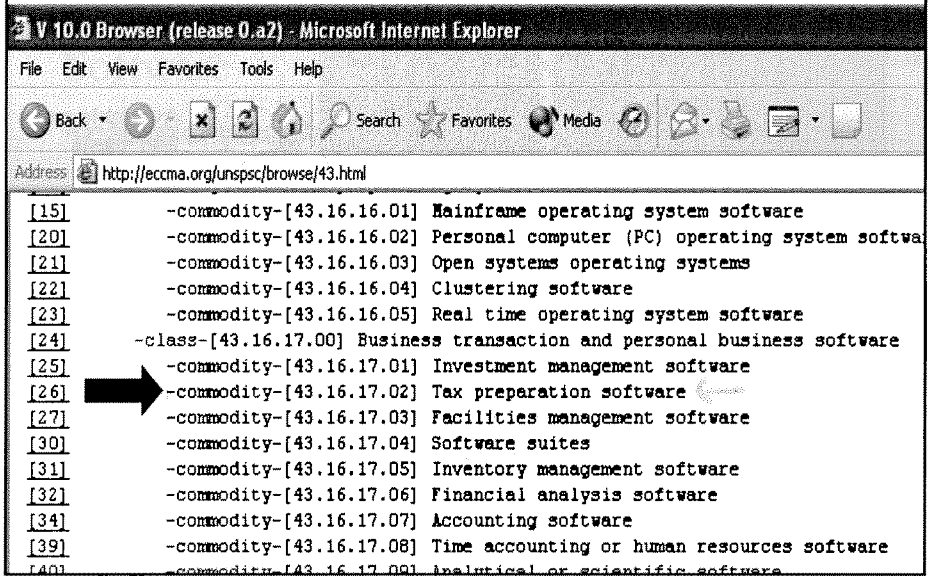
- By using standard taxonomies
- And by putting the corresponding tModelKey's keyedReferences in the category bags of services



Three Standard Taxonomies in UDDI

1. NAICS - North American Industrial Classification Scheme (Industry codes - US Govt.)
   - tModelKey="uuid:c0b9fe13-179f-413d-8a5b-5004db8e5bb2"
   - http://www.naics.com
2. UN/SPSC - Universal Standard Products and Services Classification (ECMA)
   - tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
   - http://eccma.org/unspsc
3. ISO 3166 Geographical taxonomy
   - tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"
   - http://www.iso.ch/iso/en/prods-services/iso3166ma/index.html

UNSPSC - Universal Standard Products and Services Classification


V 10.0 Browser (release 0.a2) - Microsoft Internet Explorer

Address http://eccma.org/unspsc/browse/43.html

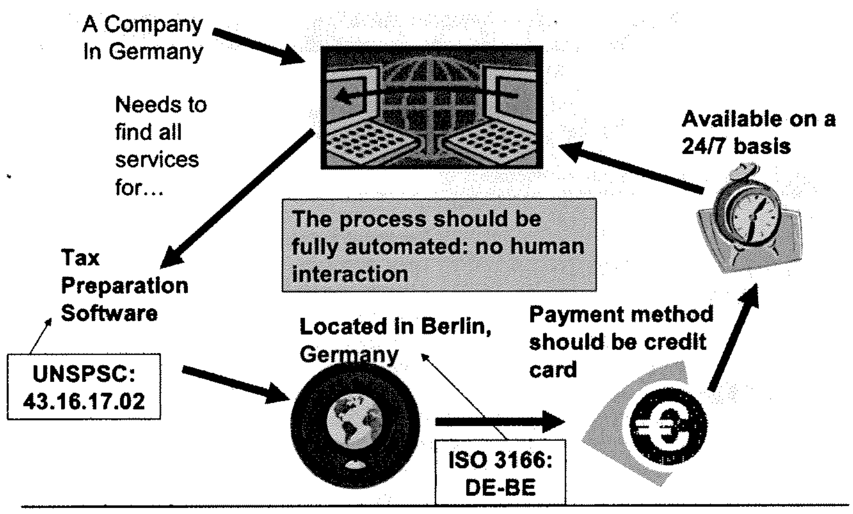
```
[15]      -commodity-[43.16.16.01] Mainframe operating system software
[20]      -commodity-[43.16.16.02] Personal computer (PC) operating system softwa
[21]      -commodity-[43.16.16.03] Open systems operating systems
[22]      -commodity-[43.16.16.04] Clustering software
[23]      -commodity-[43.16.16.05] Real time operating system software
[24]    -class-[43.16.17.00] Business transaction and personal business software
[25]      -commodity-[43.16.17.01] Investment management software
[26]      -commodity-[43.16.17.02] Tax preparation software
[27]      -commodity-[43.16.17.03] Facilities management software
[30]      -commodity-[43.16.17.04] Software suites
[31]      -commodity-[43.16.17.05] Inventory management software
[32]      -commodity-[43.16.17.06] Financial analysis software
[34]      -commodity-[43.16.17.07] Accounting software
[39]      -commodity-[43.16.17.08] Time accounting or human resources software
[40]      -commodity-[43.16.17.09] Analytical or scientific software
```

ISO 3166 Codes (Countries)

(http://www.iso.ch/iso/en/prods-services/iso3166ma/)

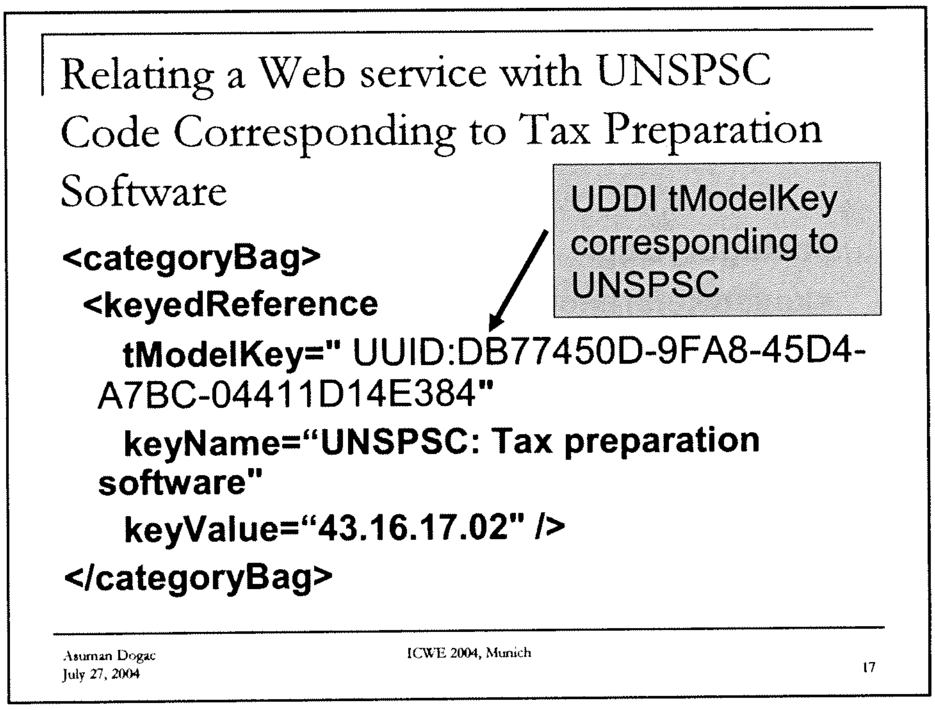
- AFGHANISTAN → AF
- ÅLAND ISLANDS → AX
- ALBANIA → AL
- ...
- GERMANY → DE
- ...
- UNITED STATES → US
- ...

Our Example


A Company In Germany
Needs to find all services for...
Tax Preparation Software
UNSPSC: 43.16.17.02
The process should be fully automated: no human interaction
Available on a 24/7 basis
Located in Berlin, Germany
ISO 3166: DE-BE
Payment method should be credit card

How to Find a Service Related with "Tax Preparation Software" in UDDI?

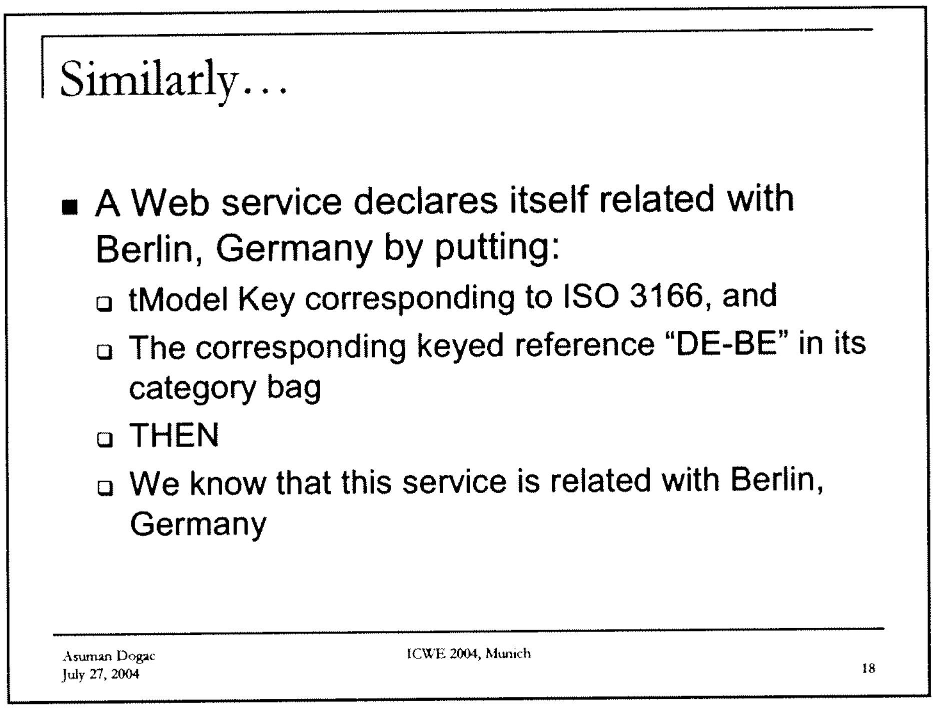
- If a service puts:
  - tModel Key corresponding to UNSPSC, and
  - The corresponding keyed reference **("43.16.17.02")** in its category bag
  - THEN
  - We know that this service is related with Tax Preparation Software



Relating a Web service with UNSPSC Code Corresponding to Tax Preparation Software

UDDI tModelKey corresponding to UNSPSC

```
<categoryBag>
  <keyedReference
    tModelKey=" UUID:DB77450D-9FA8-45D4-
   A7BC-04411D14E384"
    keyName="UNSPSC: Tax preparation
   software"
    keyValue="43.16.17.02" />
</categoryBag>
```



Similarly…

- A Web service declares itself related with Berlin, Germany by putting:
  - tModel Key corresponding to ISO 3166, and
  - The corresponding keyed reference "DE-BE" in its category bag
  - THEN
  - We know that this service is related with Berlin, Germany

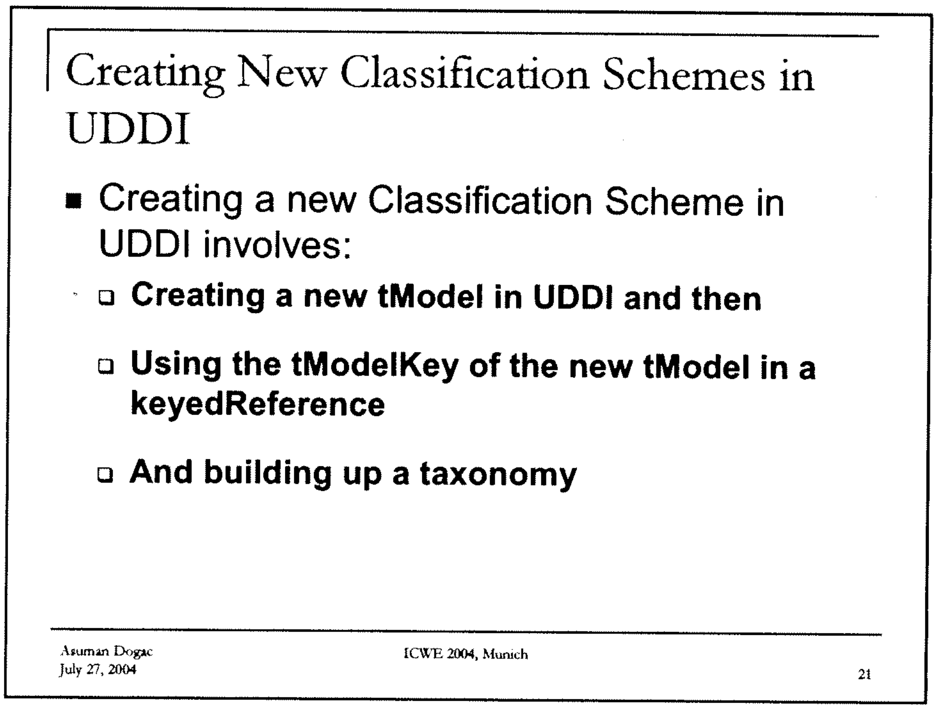
Relating a Web service with ISO 3166 Geographic Taxonomy

UDDI tModelKey
For ISO 3166

```
<categoryBag>
<keyedReference
    tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-
  0411d8d19e88"
    keyName="Berlin, Germany"
   keyValue="DE-BE" />
</categoryBag>
```

We Need More Semantics…

- How about the other properties of service we are looking for?
- Payment method? Delivery time? Service availability?
- We can create new classification schemes in UDDI



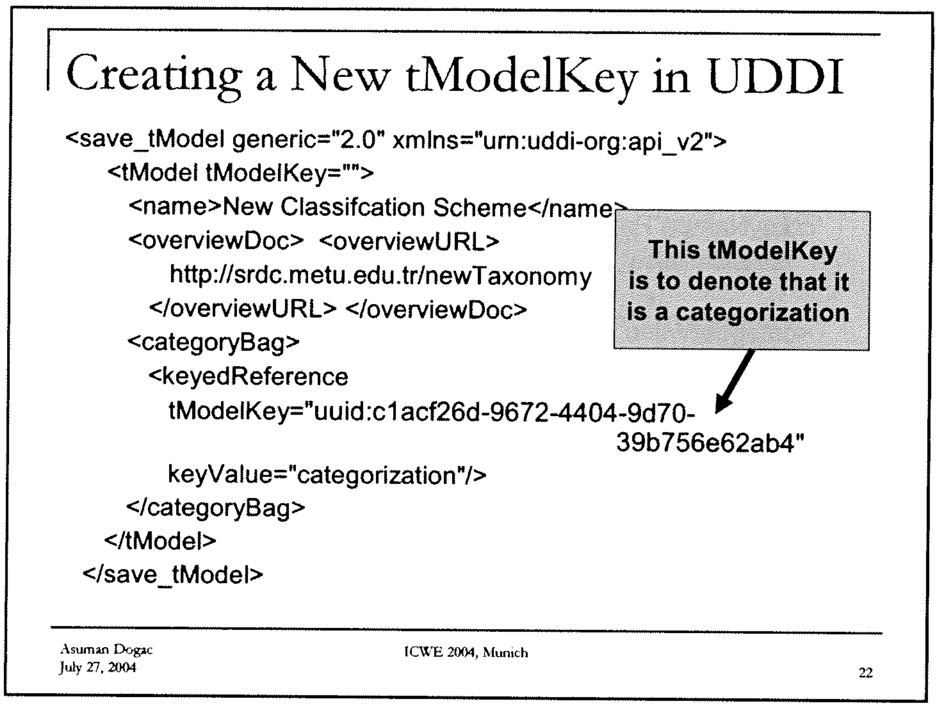
Creating New Classification Schemes in UDDI

- Creating a new Classification Scheme in UDDI involves:
  - **Creating a new tModel in UDDI and then**
  - **Using the tModelKey of the new tModel in a keyedReference**
  - **And building up a taxonomy**

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

21



Creating a New tModelKey in UDDI

```
<save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModel tModelKey="">
      <name>New Classifcation Scheme</name>
      <overviewDoc>  <overviewURL>
          http://srdc.metu.edu.tr/newTaxonomy
        </overviewURL> </overviewDoc>
      <categoryBag>
        <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-
                                              39b756e62ab4"
          keyValue="categorization"/>
      </categoryBag>
    </tModel>
  </save_tModel>
```

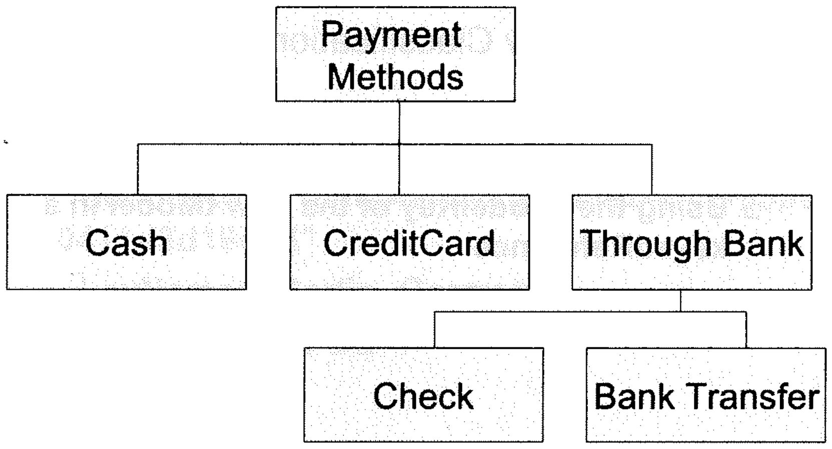
This tModelKey is to denote that it is a categorization

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

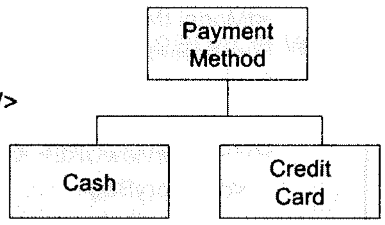
22

Creating a New Taxonomy for Payment Methods in UDDI

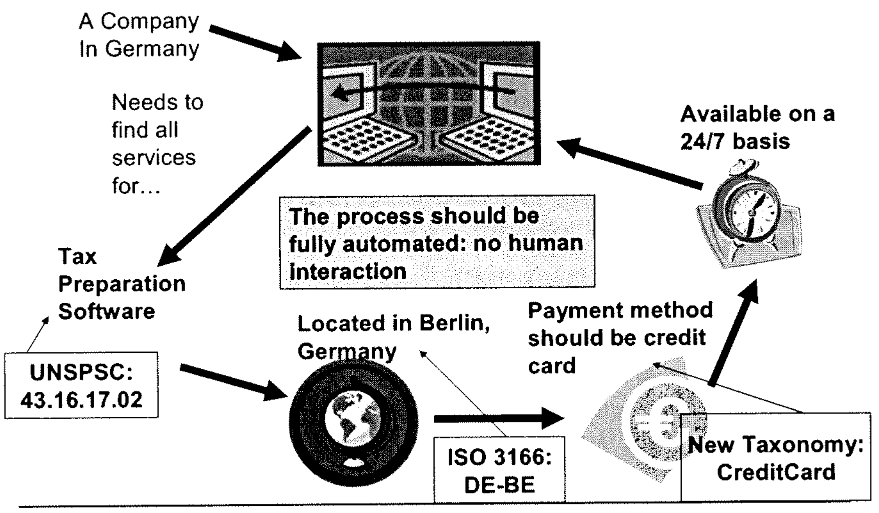


Creating a New Taxonomy

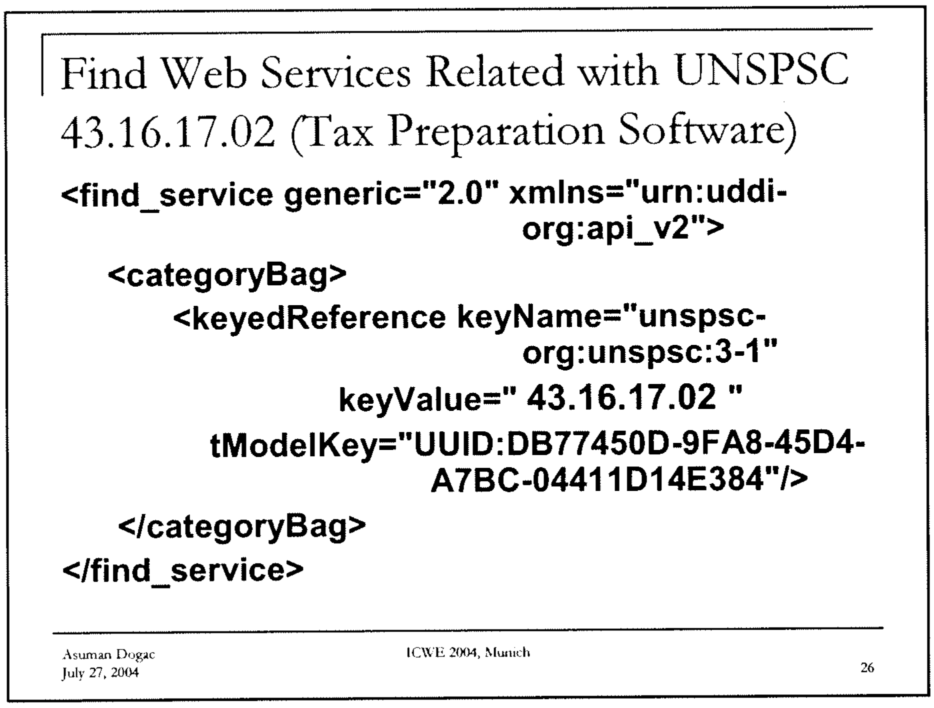
```
<categoryValue  keyValue="0"
  keyName="Payment Methods"
  isValid="false"  parentKeyValue=""/>

<categoryValue  keyValue="1"
  keyName="Cash"
   isValid="false" parentKeyValue="0"/>

<categoryValue keyValue="2"
  keyName="CreditCard"
  isValid="true" parentKeyValue="0"/>
```



Our Example


A Company In Germany
Needs to find all services for…
Available on a 24/7 basis
The process should be fully automated: no human interaction
Tax Preparation Software
UNSPSC: 43.16.17.02
Located in Berlin, Germany
Payment method should be credit card
ISO 3166: DE-BE
New Taxonomy: CreditCard

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
25



Find Web Services Related with UNSPSC 43.16.17.02 (Tax Preparation Software)

```
<find_service generic="2.0" xmlns="urn:uddi-
                              org:api_v2">
   <categoryBag>
       <keyedReference keyName="unspsc-
                              org:unspsc:3-1"
                 keyValue=" 43.16.17.02 "
          tModelKey="UUID:DB77450D-9FA8-45D4-
                          A7BC-04411D14E384"/>
    </categoryBag>
</find_service>
```

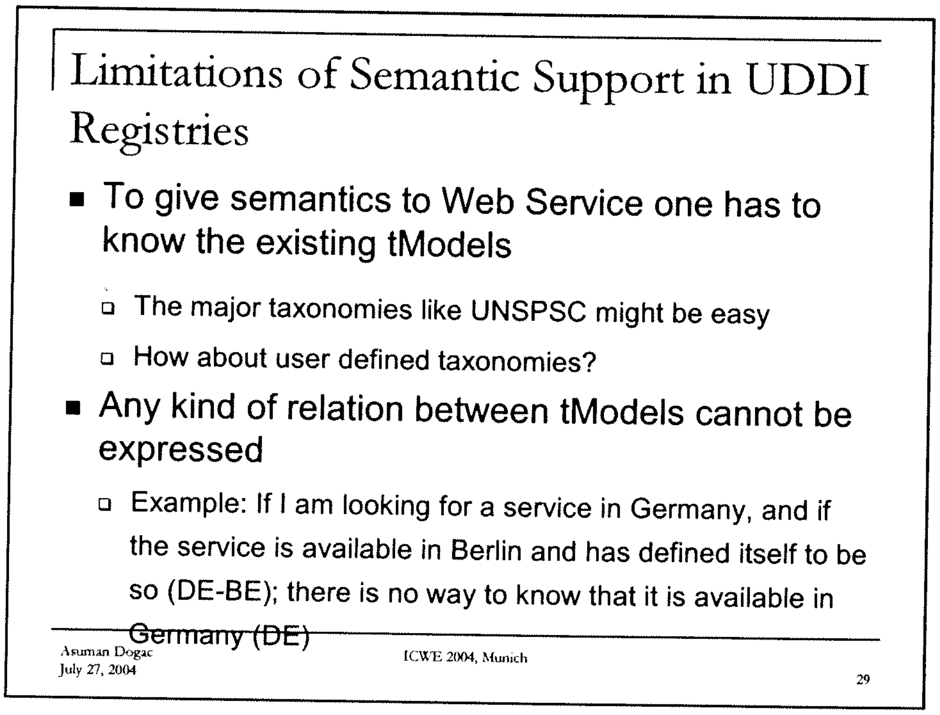
Asuman Dogac
July 27, 2004
ICWE 2004, Munich
26

A Web Service can be searched with any number of Keyed References

- Use UNSPSC for "Tax Preparation Software"
- Use ISO 3166 for "Berlin, Germany"
- Use the newly created taxonomy to relate this service with "CreditCard"
- By default, all "keyedReference" elements passed in a bag have logical "AND" among them
- This default "AND" can be overridden by "orAllKeys findQualifyer"

Example

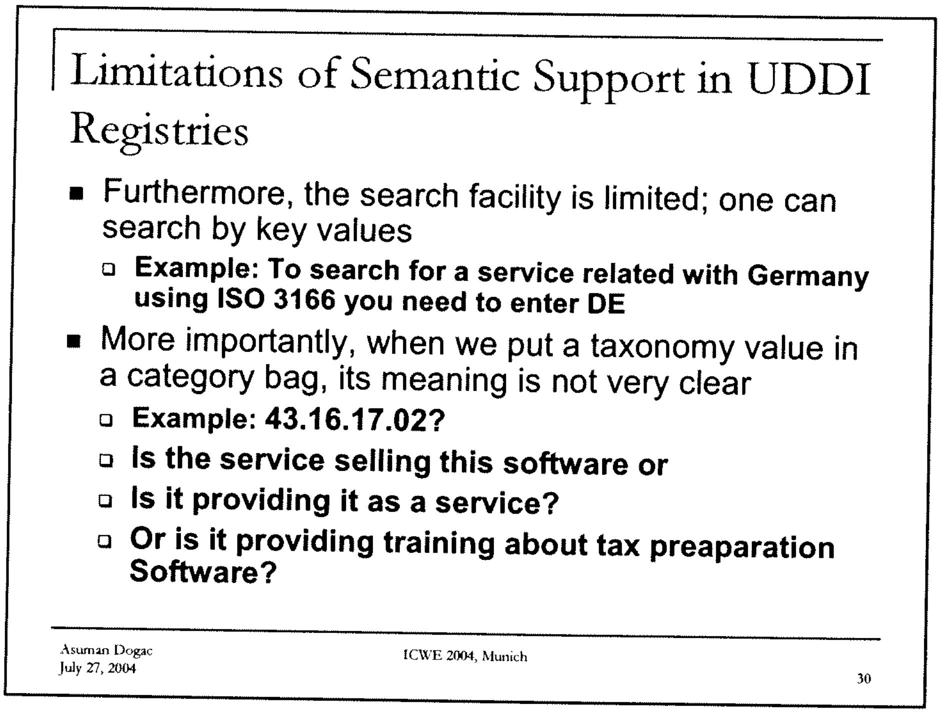
```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <find_service xmlns="urn:uddi-org:api_v2" generic="2.0">
      <findQualifiers>
        <findQualifier>orAllKeys</findQualifier>
      </findQualifiers>
<categoryBag>
        <keyedReference keyValue=" 43.16.17.02 "
         tModelKey=" UUID:DB77450D-9FA8-45D4-A7BC-
   04411D14E384 " />
        <keyedReference keyValue="DE-BE"
   tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88" />
      </categoryBag>
    </find_service>
  </Body>
</Envelope>
```



Limitations of Semantic Support in UDDI Registries

- To give semantics to Web Service one has to know the existing tModels
  - The major taxonomies like UNSPSC might be easy
  - How about user defined taxonomies?
- Any kind of relation between tModels cannot be expressed
  - Example: If I am looking for a service in Germany, and if the service is available in Berlin and has defined itself to be so (DE-BE); there is no way to know that it is available in Germany (DE)

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
29



Limitations of Semantic Support in UDDI Registries

- Furthermore, the search facility is limited; one can search by key values
  - **Example: To search for a service related with Germany using ISO 3166 you need to enter DE**
- More importantly, when we put a taxonomy value in a category bag, its meaning is not very clear
  - **Example: 43.16.17.02?**
  - **Is the service selling this software or**
  - **Is it providing it as a service?**
  - **Or is it providing training about tax preaparation Software?**

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
30

Why UDDI Semantics is Limited?

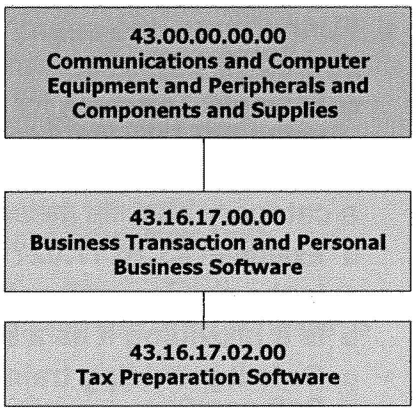
- **Putting a keyed reference into the category bag of a service merely says that service is somehow related with this value: but does not say anything on HOW!**
- **You can not directly assign named properties to the services through UDDI!**
- **A taxonomy is a hierarchy and a unique code is usually assigned to each node of the hierarchy: no properties here either!**
- **And UDDI uses taxonomies to describe the semantic of Web services by relating them to tModels**

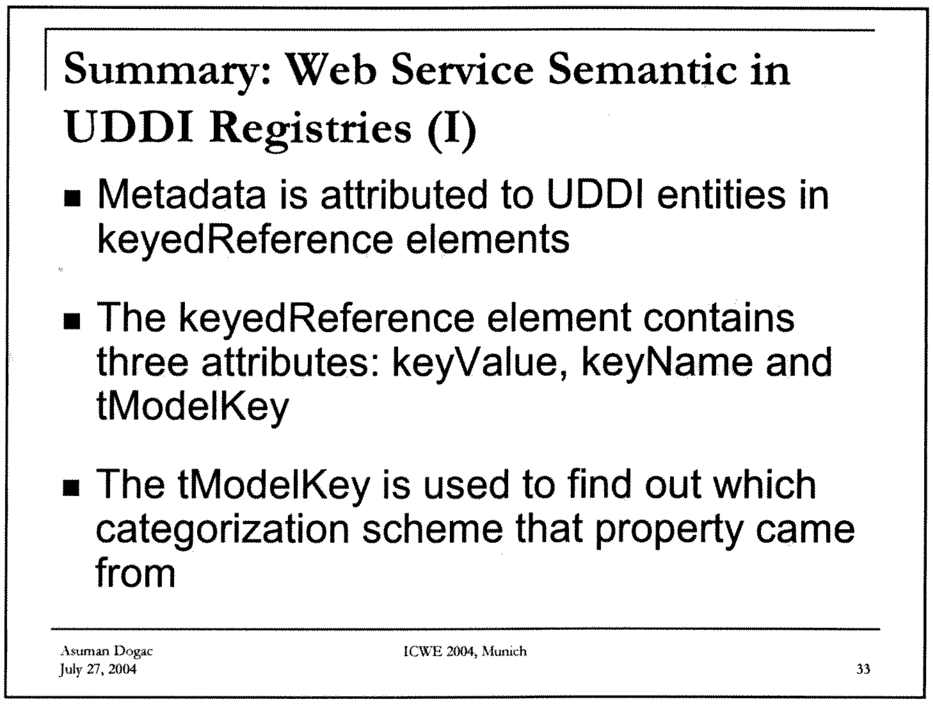
Taxonomies Define Only Class/Subclass Relationship: An Example Taxonomy: UNSPSC

Through taxonomies:

• It is not possible to define properties of services

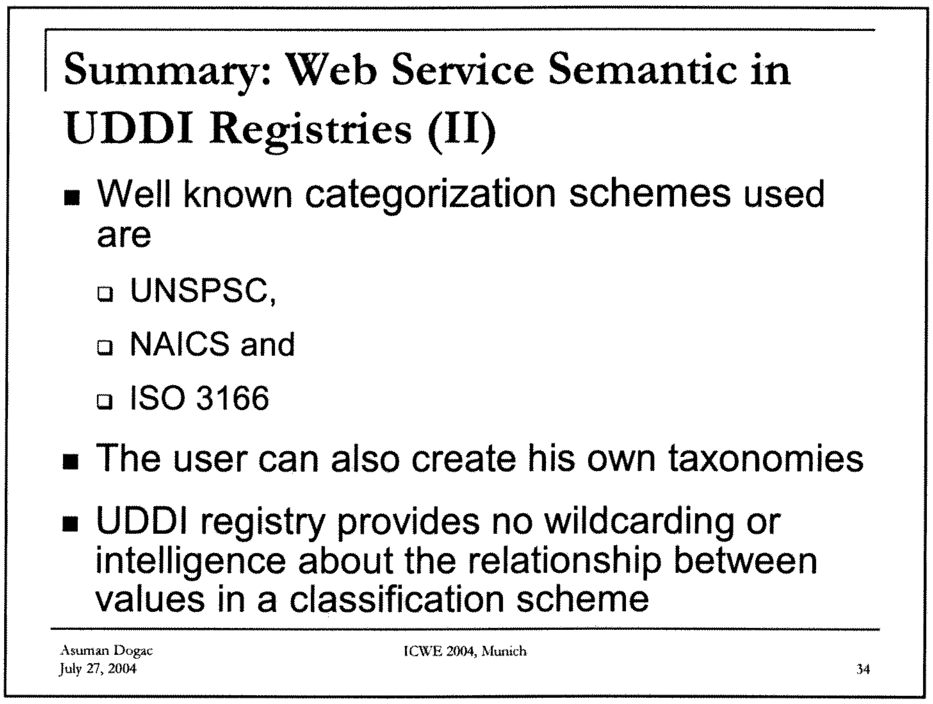
• It is not possible to relate service classes with one another


43.00.00.00.00
Communications and Computer Equipment and Peripherals and Components and Supplies
43.16.17.00.00
Business Transaction and Personal Business Software
43.16.17.02.00
Tax Preparation Software



Summary: Web Service Semantic in UDDI Registries (I)

- Metadata is attributed to UDDI entities in keyedReference elements
- The keyedReference element contains three attributes: keyValue, keyName and tModelKey
- The tModelKey is used to find out which categorization scheme that property came from



Summary: Web Service Semantic in UDDI Registries (II)

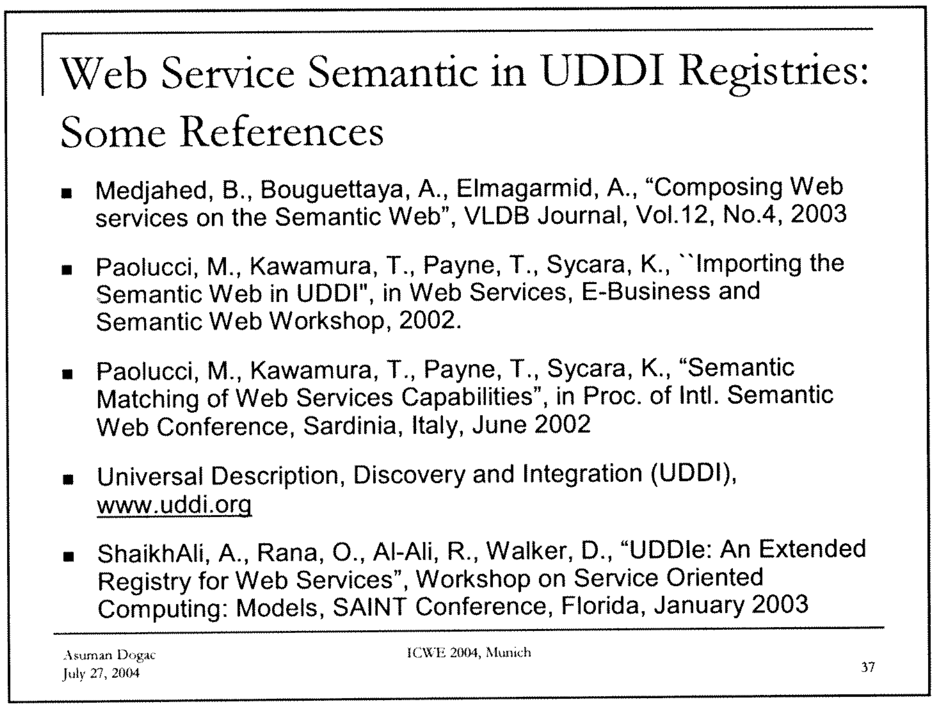
- Well known categorization schemes used are
  - UNSPSC,
  - NAICS and
  - ISO 3166
- The user can also create his own taxonomies
- UDDI registry provides no wildcarding or intelligence about the relationship between values in a classification scheme

Summary: Web Service Semantic in UDDI Registries (III)

- UDDI provides limited semantics for Web Services
- Taxonomies are not enough to describe semantics
- Ontologies are needed
- So it is time to investigate ontologies!

Web Service Semantic in UDDI Registries: Some References

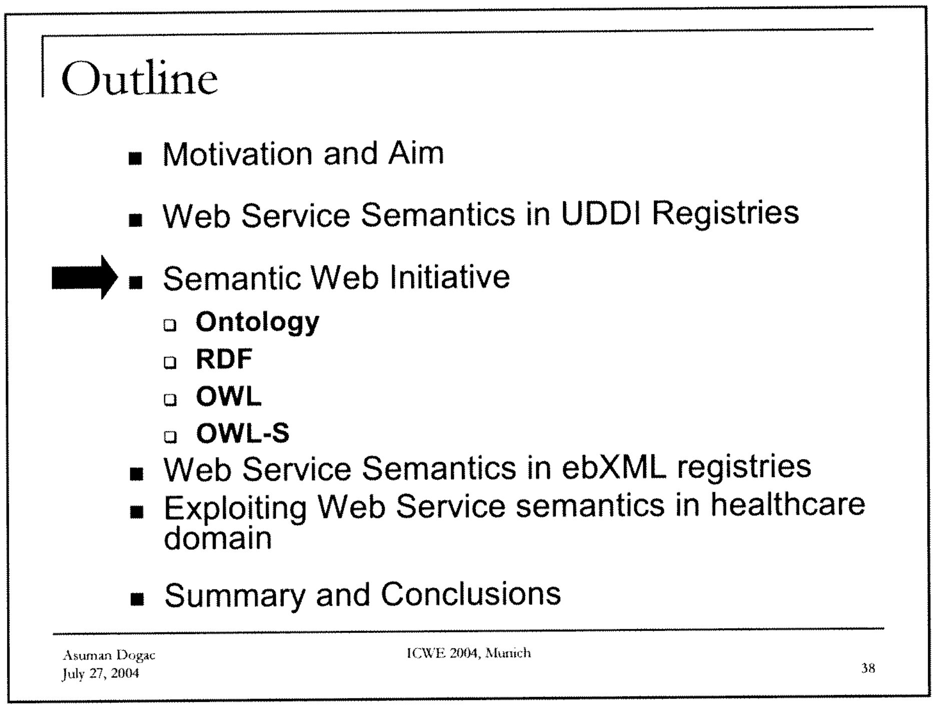
- Bussler, C., Fensel, D., Maedche, A., “A Conceptual Architecture for Semantic Web Enabled Web Service”, ACM Sigmod Record, Vol. 31, No. 4, December 2002
- Dogac, A., Cingil, I., Laleci, G. B., Kabak, Y., “Improving the Functionality of UDDI Registries through Web Service Semantics”, 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 2002
- 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
- Januszewski, K, “The Importance of Metadata: Reification, Categorization, and UDDI”, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuddi/html/impmetadata.asp
- McIlraith, S. A., Martin, D. L., “Bringing Semantics to Web Services”, IEEE Intelligent Systems, Vol.18, No.1, 2003



Web Service Semantic in UDDI Registries: Some References

- Medjahed, B., Bouguettaya, A., Elmagarmid, A., "Composing Web services on the Semantic Web", VLDB Journal, Vol.12, No.4, 2003
- Paolucci, M., Kawamura, T., Payne, T., Sycara, K., ``Importing the Semantic Web in UDDI", in Web Services, E-Business and Semantic Web Workshop, 2002.
- Paolucci, M., Kawamura, T., Payne, T., Sycara, K., "Semantic Matching of Web Services Capabilities", in Proc. of Intl. Semantic Web Conference, Sardinia, Italy, June 2002
- Universal Description, Discovery and Integration (UDDI), www.uddi.org
- ShaikhAli, A., Rana, O., Al-Ali, R., Walker, D., "UDDIe: An Extended Registry for Web Services", Workshop on Service Oriented Computing: Models, SAINT Conference, Florida, January 2003

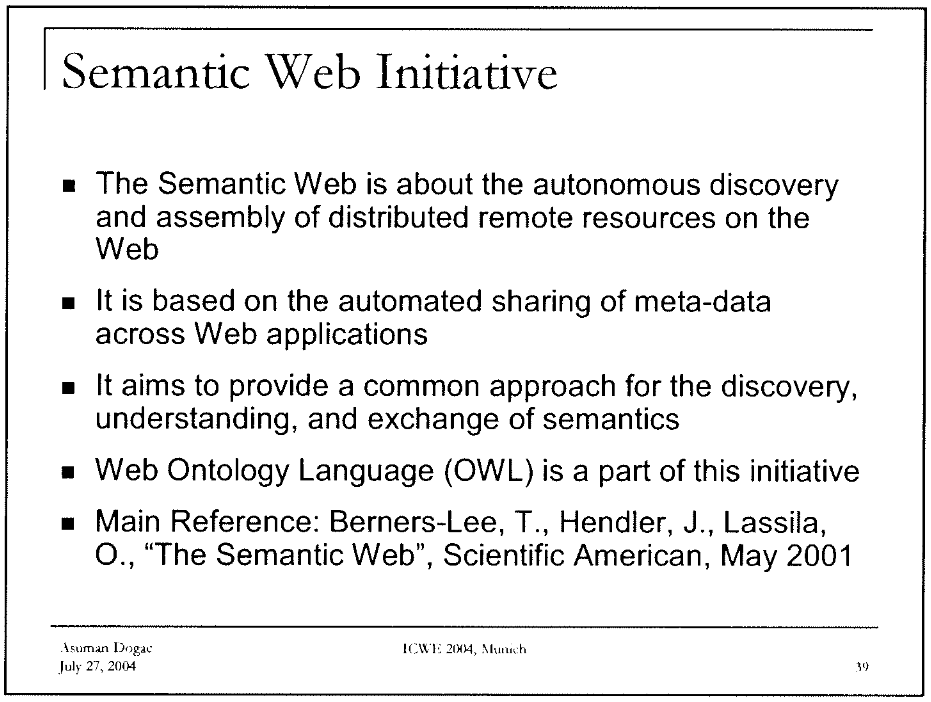
Asuman Dogac
July 27, 2004
ICWE 2004, Munich
37



Outline

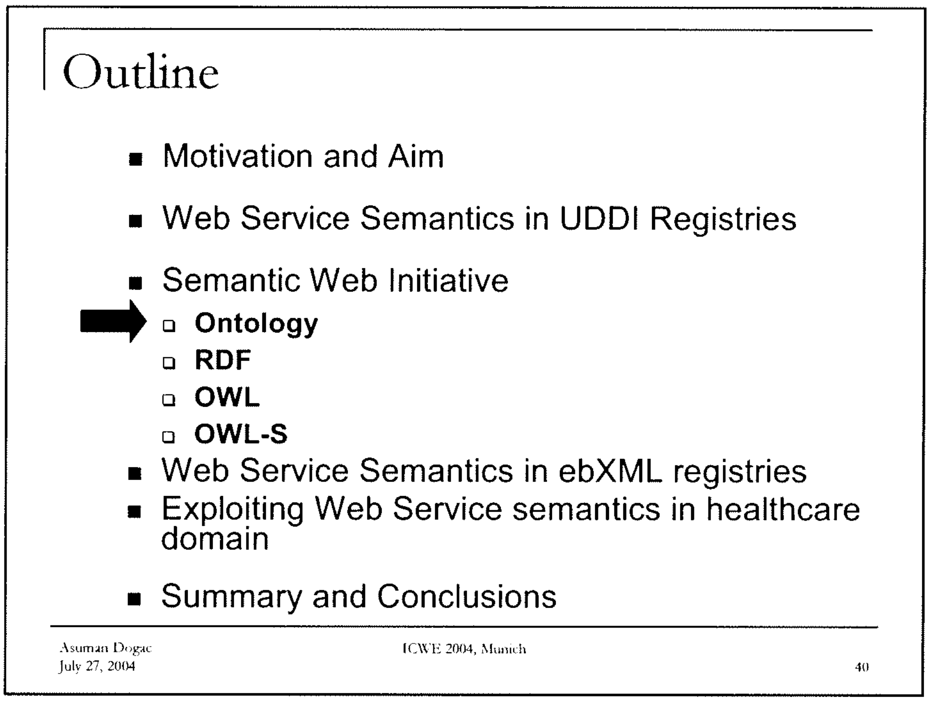
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- ➡ Semantic Web Initiative
  - **Ontology**
  - **RDF**
  - **OWL**
  - **OWL-S**
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
38



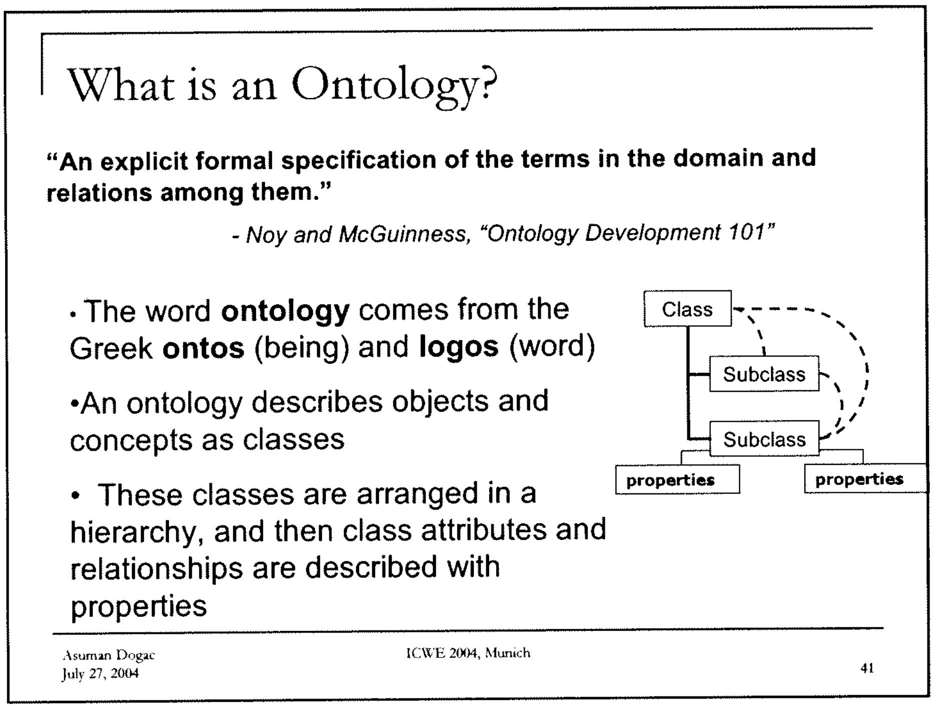
Semantic Web Initiative

- The Semantic Web is about the autonomous discovery and assembly of distributed remote resources on the Web
- It is based on the automated sharing of meta-data across Web applications
- It aims to provide a common approach for the discovery, understanding, and exchange of semantics
- Web Ontology Language (OWL) is a part of this initiative
- Main Reference: Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001



Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
  - ➡ **Ontology**
  - **RDF**
  - **OWL**
  - **OWL-S**
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions

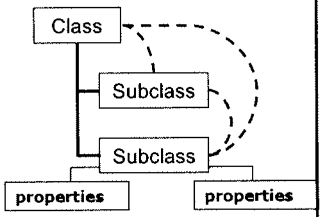


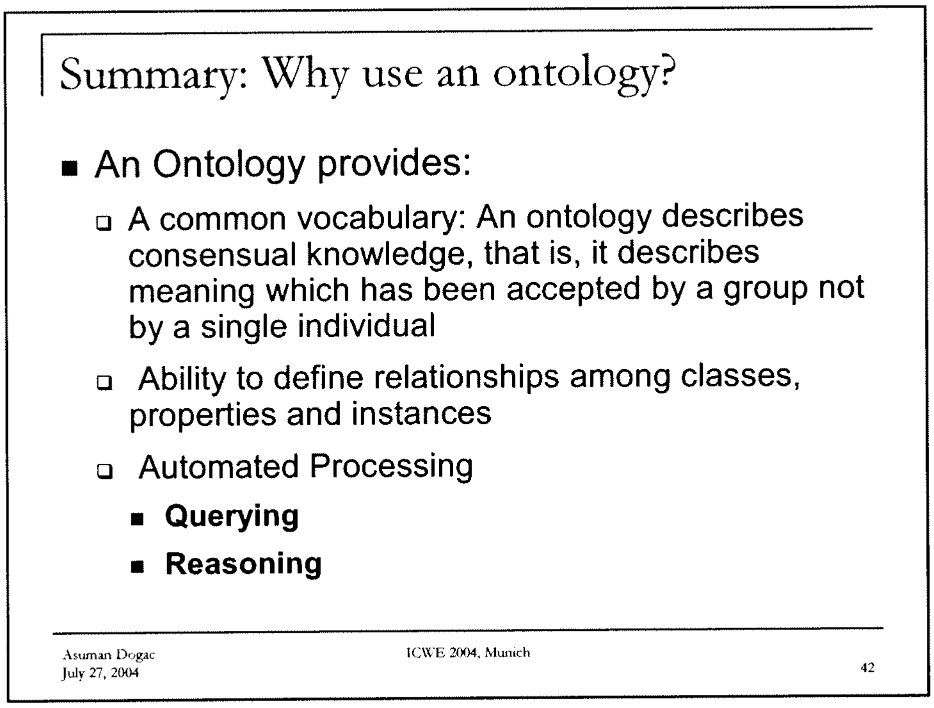
What is an Ontology?

"An explicit formal specification of the terms in the domain and relations among them."

- Noy and McGuinness, "Ontology Development 101"

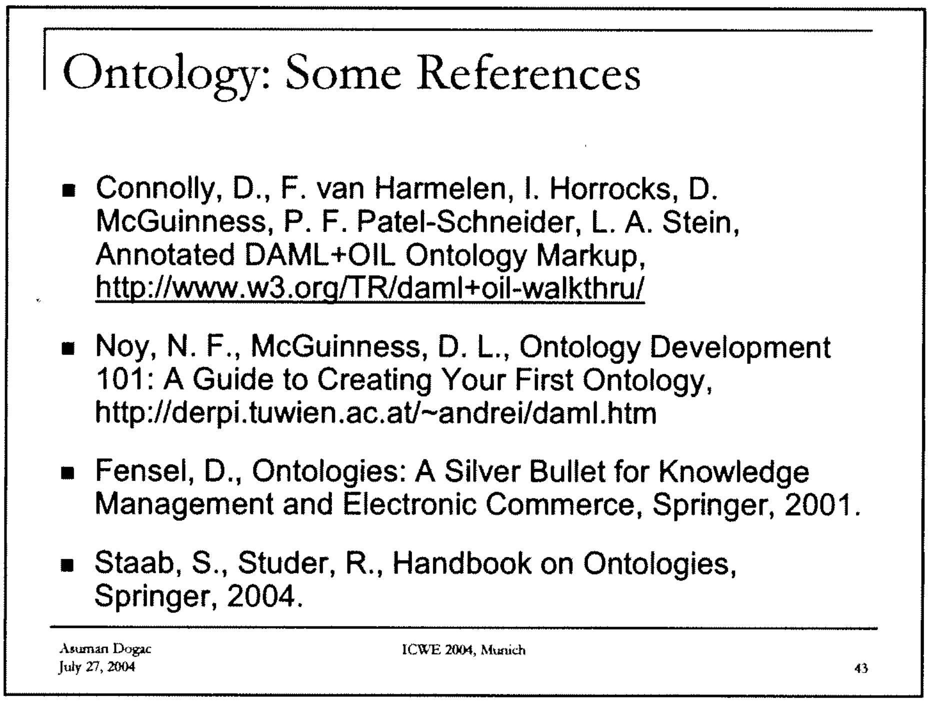
- The word **ontology** comes from the Greek **ontos** (being) and **logos** (word)
- An ontology describes objects and concepts as classes
- These classes are arranged in a hierarchy, and then class attributes and relationships are described with properties





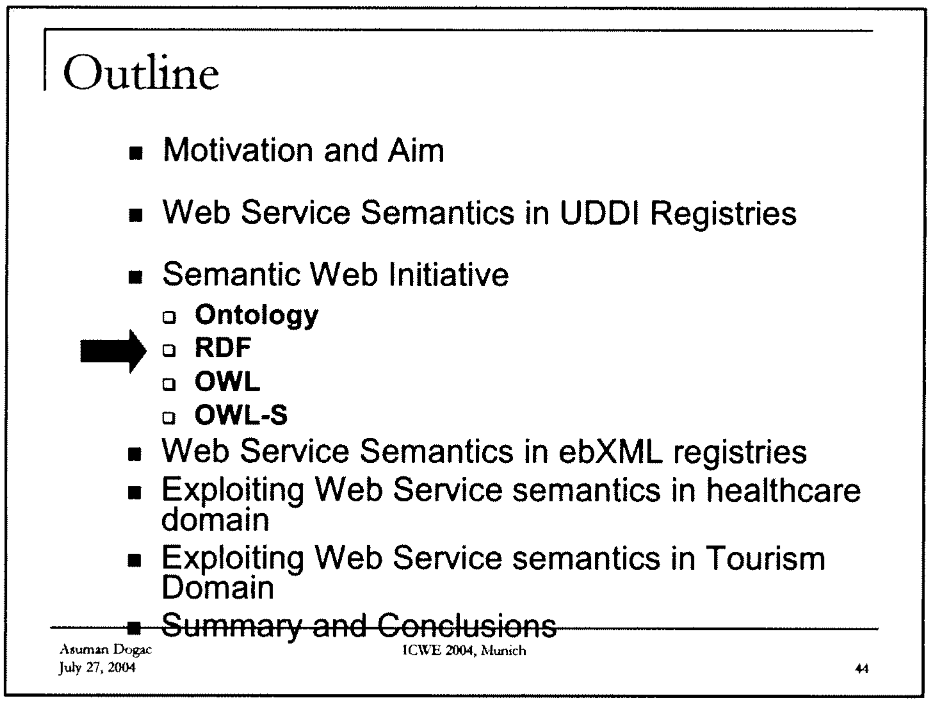
Summary: Why use an ontology?

- An Ontology provides:
  - A common vocabulary: An ontology describes consensual knowledge, that is, it describes meaning which has been accepted by a group not by a single individual
  - Ability to define relationships among classes, properties and instances
  - Automated Processing
    - **Querying**
    - **Reasoning**



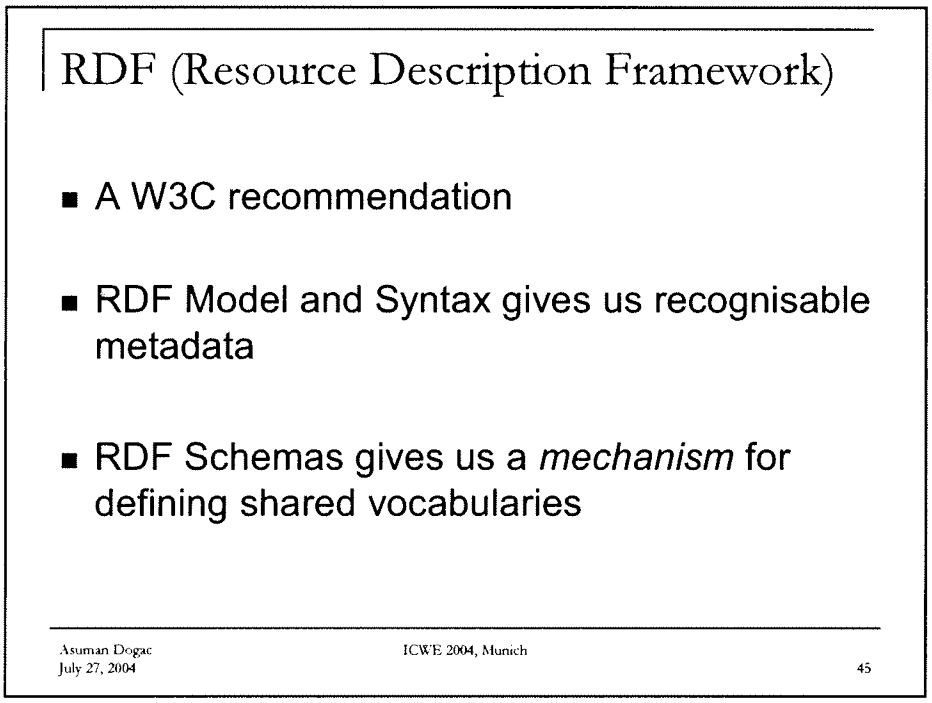
Ontology: Some References

- Connolly, D., F. van Harmelen, I. Horrocks, D. McGuinness, P. F. Patel-Schneider, L. A. Stein, Annotated DAML+OIL Ontology Markup, http://www.w3.org/TR/daml+oil-walkthru/
- Noy, N. F., McGuinness, D. L., Ontology Development 101: A Guide to Creating Your First Ontology, http://derpi.tuwien.ac.at/~andrei/daml.htm
- Fensel, D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.
- Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.



Outline

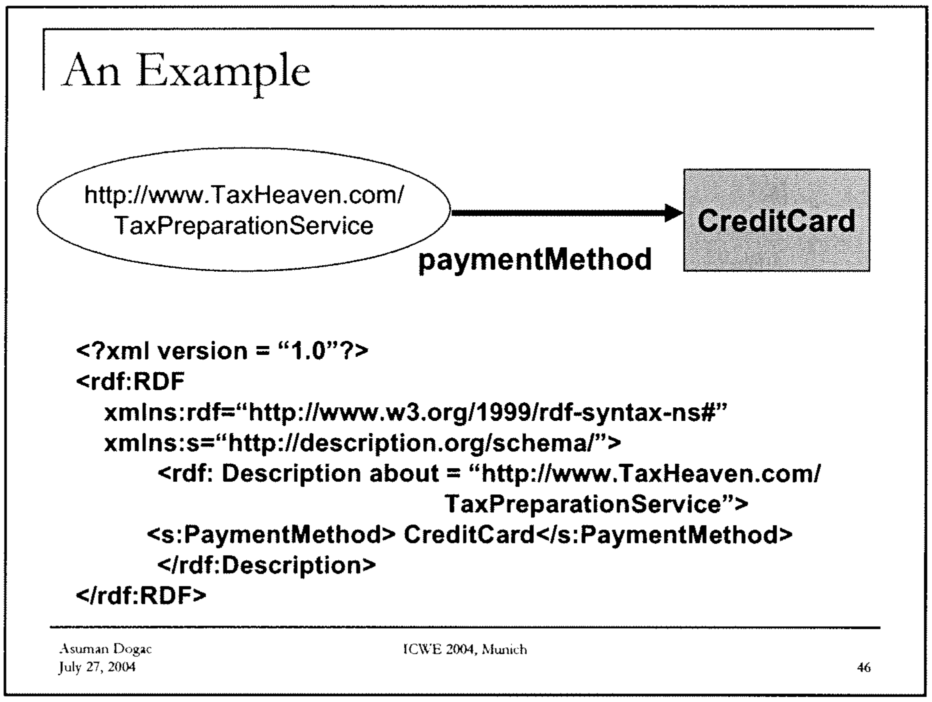
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
  - **Ontology**
  - ➡ **RDF**
  - **OWL**
  - **OWL-S**
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Exploiting Web Service semantics in Tourism Domain
- Summary and Conclusions



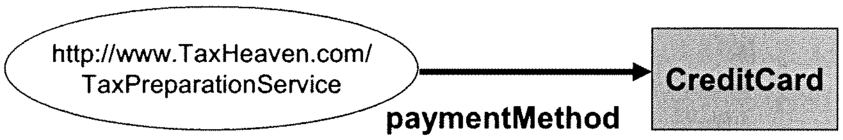
RDF (Resource Description Framework)

- A W3C recommendation
- RDF Model and Syntax gives us recognisable metadata
- RDF Schemas gives us a *mechanism* for defining shared vocabularies

Asuman Dogac
July 27, 2004 ICWE 2004, Munich 45



An Example



```
<?xml version = "1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
      <rdf: Description about = "http://www.TaxHeaven.com/
                               TaxPreparationService">
     <s:PaymentMethod> CreditCard</s:PaymentMethod>
      </rdf:Description>
</rdf:RDF>
```

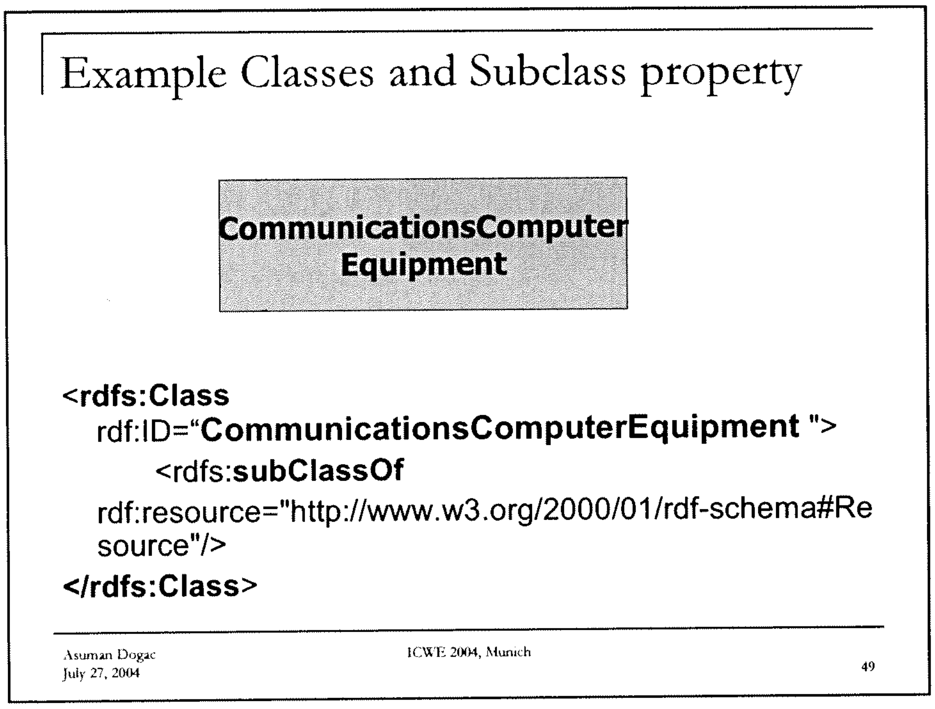
Asuman Dogac
July 27, 2004 ICWE 2004, Munich 46

RDF Core Classes

- **rdfs:Resource** - All things being described by RDF expressions are resources and are considered to be instances of the class rdfs:Resource
- **rdfs:Class** - represents the generic concept of a type or category and can be defined to represent almost everything, e.g. Web pages, people, document types…
- **rdf:Property** - represents the subset of RDF resources that are properties

RDF Core Properties

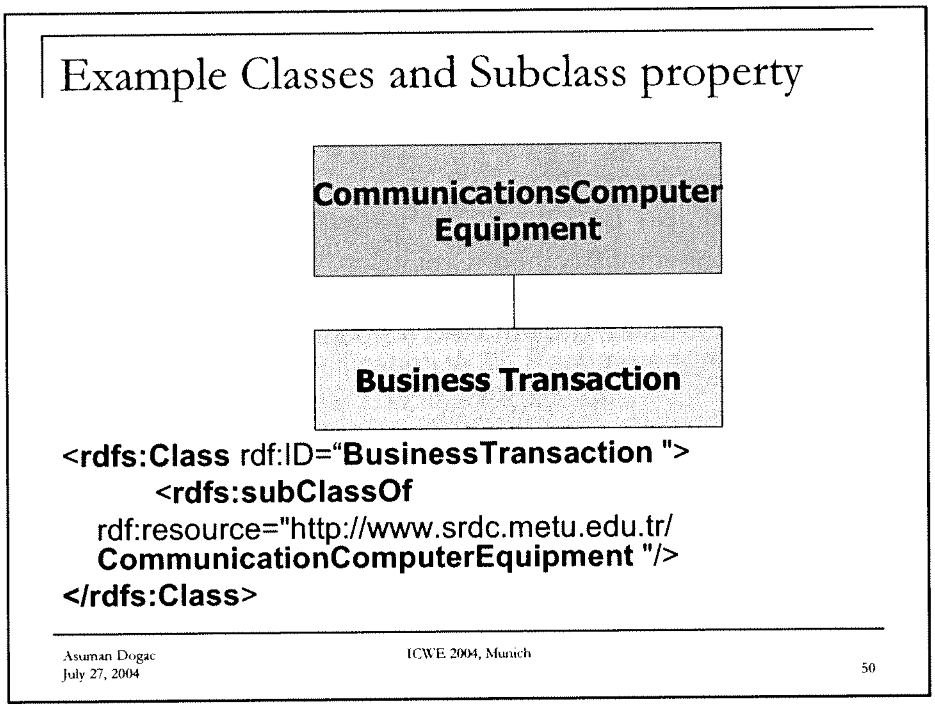
- **rdfs:subClassOf -** This property specifies a subset/superset relation between classes
- **rdfs:subPropertyOf -** is an instance of rdf:Property that is used to specify that one property is a specialization of another
- **rdfs: range** - is used to define that the values of a property are instances of one or more stated classes
- **rdfs: domain** - is used to state that any resource that has a given property is an instance of one or more classes



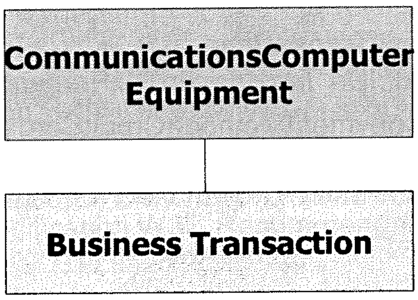
Example Classes and Subclass property

CommunicationsComputer Equipment

```
<rdfs:Class
  rdf:ID="CommunicationsComputerEquipment ">
      <rdfs:subClassOf
  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Re
  source"/>
</rdfs:Class>
```

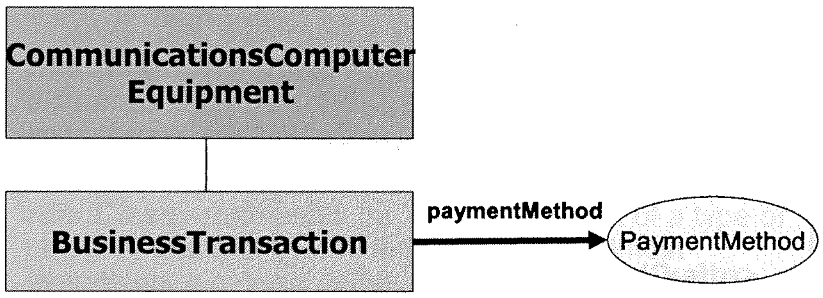


Example Classes and Subclass property



```
<rdfs:Class rdf:ID="BusinessTransaction ">
       <rdfs:subClassOf
  rdf:resource="http://www.srdc.metu.edu.tr/
  CommunicationComputerEquipment "/>
</rdfs:Class>
```

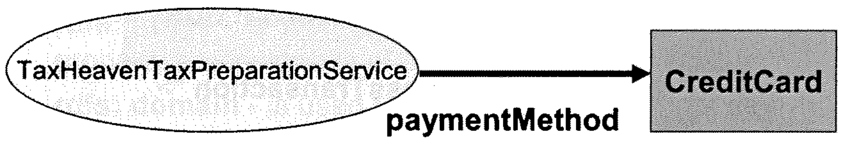
RDF Property Example

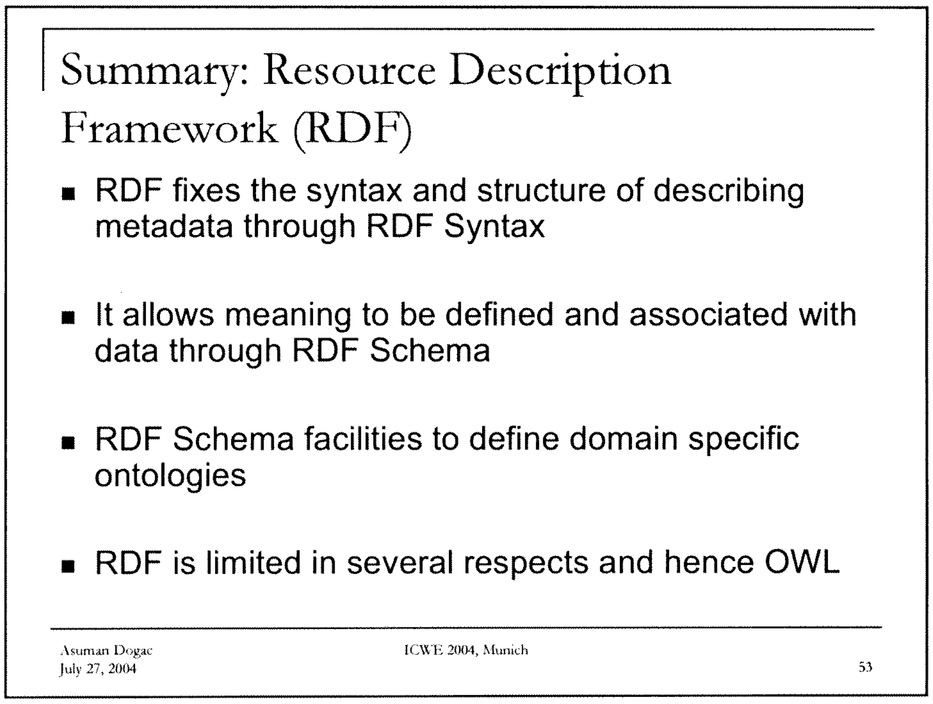


```
<rdfs:Property rdf:ID="paymentMethod">
        <rdfs:domain rdf:resource="#BusinessTransaction"/>
        <rdfs:range
   rdf:resource="http://www.srdc.metu.edu.tr/BusinessOnt#PaymentMeth
   od"/>
   </rdfs:Property>
```

An Example Class Instance in RDF

```
<BusinessTransaction
        rdf:ID="TaxHeavenTaxPreparationService">
        <paymentMethod>CreditCard</paymentMethod>
</BusinessTransaction >
```

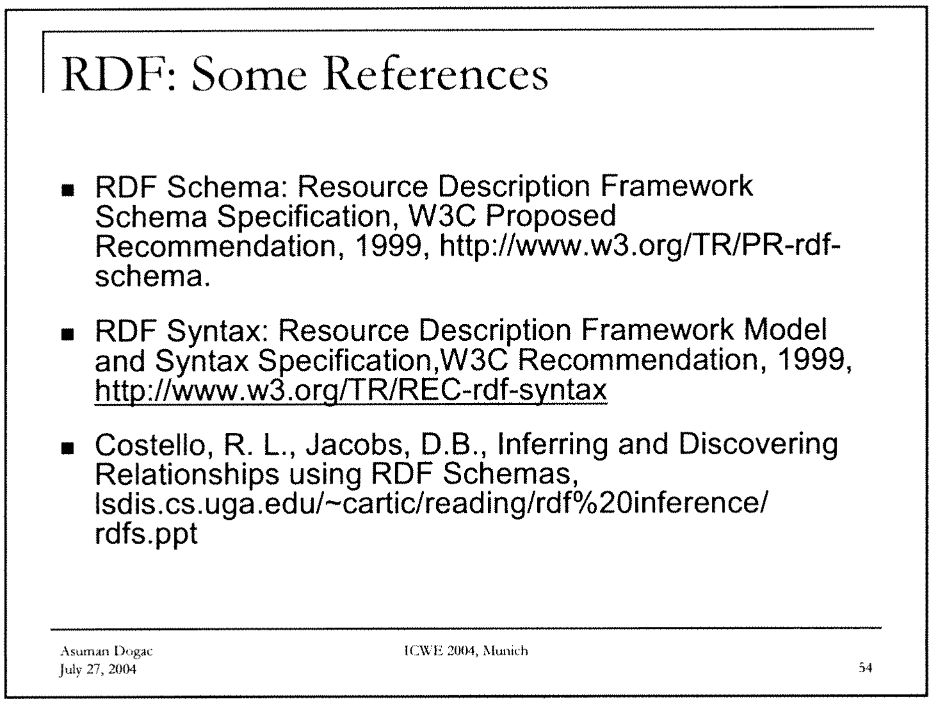




Summary: Resource Description Framework (RDF)

- RDF fixes the syntax and structure of describing metadata through RDF Syntax
- It allows meaning to be defined and associated with data through RDF Schema
- RDF Schema facilities to define domain specific ontologies
- RDF is limited in several respects and hence OWL

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
53



RDF: Some References

- RDF Schema: Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.
- RDF Syntax: Resource Description Framework Model and Syntax Specification,W3C Recommendation, 1999, http://www.w3.org/TR/REC-rdf-syntax
- Costello, R. L., Jacobs, D.B., Inferring and Discovering Relationships using RDF Schemas, lsdis.cs.uga.edu/~cartic/reading/rdf%20inference/ rdfs.ppt

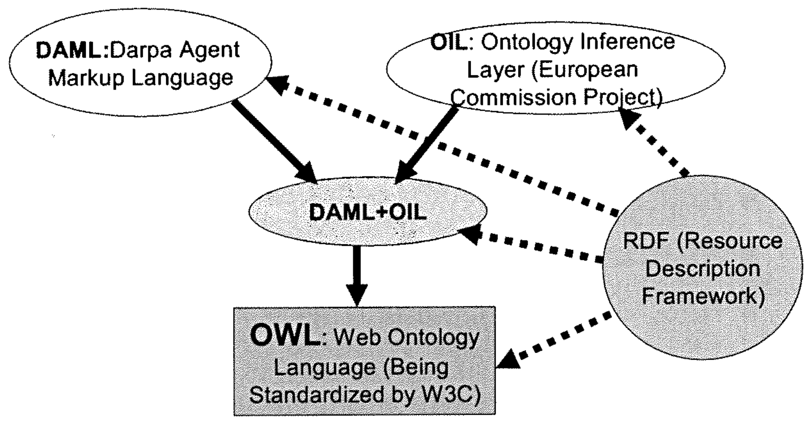
Asuman Dogac
July 27, 2004
ICWE 2004, Munich
54

Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
  - **Ontology**
  - **RDF**
  - **OWL**
  - **OWL-S**
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions

Web Ontology Language: OWL

Ontology Languages and OWL


DAML:Darpa Agent Markup Language
OIL: Ontology Inference Layer (European Commission Project)
DAML+OIL
RDF (Resource Description Framework)
OWL: Web Ontology Language (Being Standardized by W3C)

OWL = RDF Schema + more

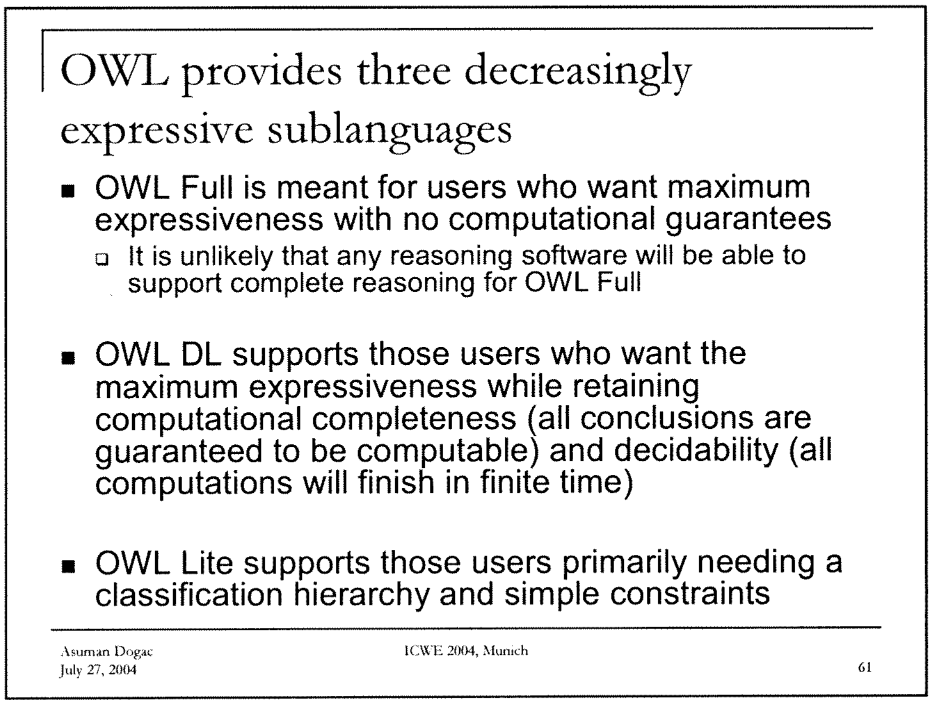
- All of the elements/attributes provided by RDF and RDF Schema can be used when creating an OWL document
- OWL classes permit much greater expressiveness than RDF Schema classes
- Consequently, OWL has created their own Class, owl:Class

RDF Schema Features used in OWL

- *rdfs:Class*
- *rdf:Property*
- *rdfs:subClassOf*
- *rdfs:subPropertyOf*
- *rdfs:domain*
- *rdfs:range*

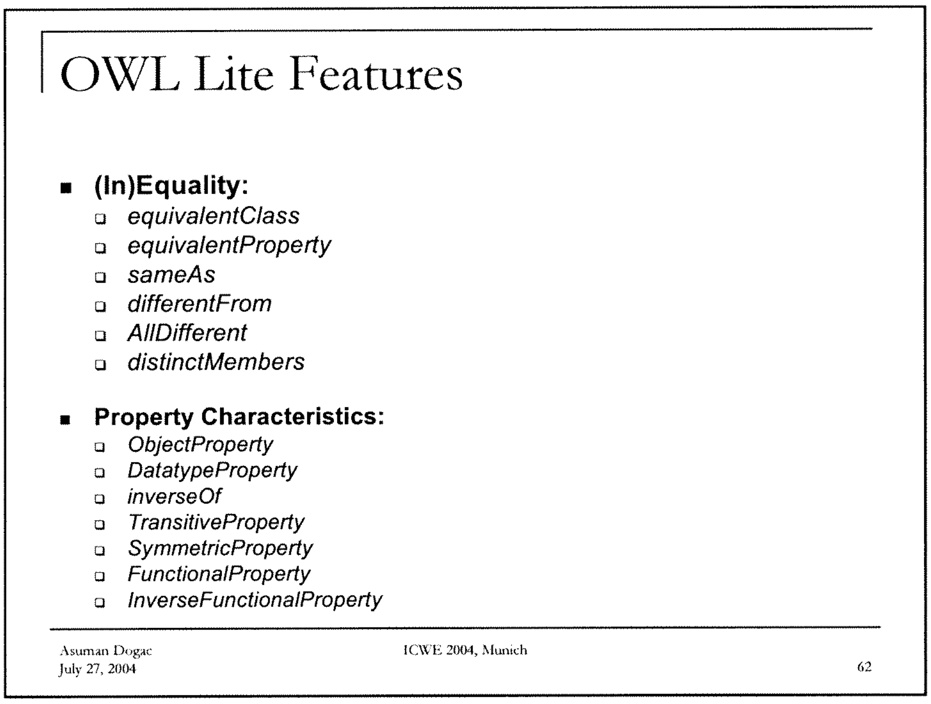
Defining Property Characteristics

- RDF Schema provides three ways to characterize a property:
  - range: use this to indicate the range of values for a property.
  - domain: use this to associate a property with a class.
  - subPropertyOf: use this to specialize a property
- OWL documents also use rdfs:range, rdfs:domain, and rdfs:subPropertyOf
- OWL has more property types that are useful in inferencing!



OWL provides three decreasingly expressive sublanguages

- OWL Full is meant for users who want maximum expressiveness with no computational guarantees
  - It is unlikely that any reasoning software will be able to support complete reasoning for OWL Full
- OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time)
- OWL Lite supports those users primarily needing a classification hierarchy and simple constraints



OWL Lite Features

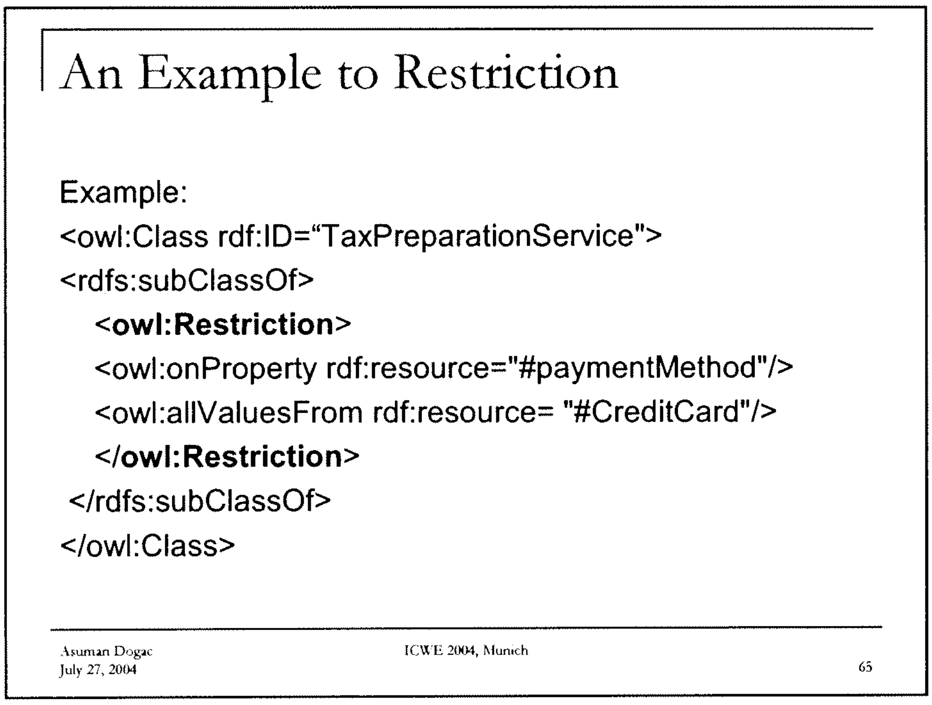
- **(In)Equality:**
  - *equivalentClass*
  - *equivalentProperty*
  - *sameAs*
  - *differentFrom*
  - *AllDifferent*
  - *distinctMembers*
- **Property Characteristics:**
  - *ObjectProperty*
  - *DatatypeProperty*
  - *inverseOf*
  - *TransitiveProperty*
  - *SymmetricProperty*
  - *FunctionalProperty*
  - *InverseFunctionalProperty*

OWL Lite Features

- **Property Type Restrictions:**
  - *Restriction*
  - *onProperty*
  - *allValuesFrom*
  - *someValuesFrom*
- **Class Intersection:**
  - *intersectionOf*

OWL Full Language constructs that are in addition to those of OWL Lite

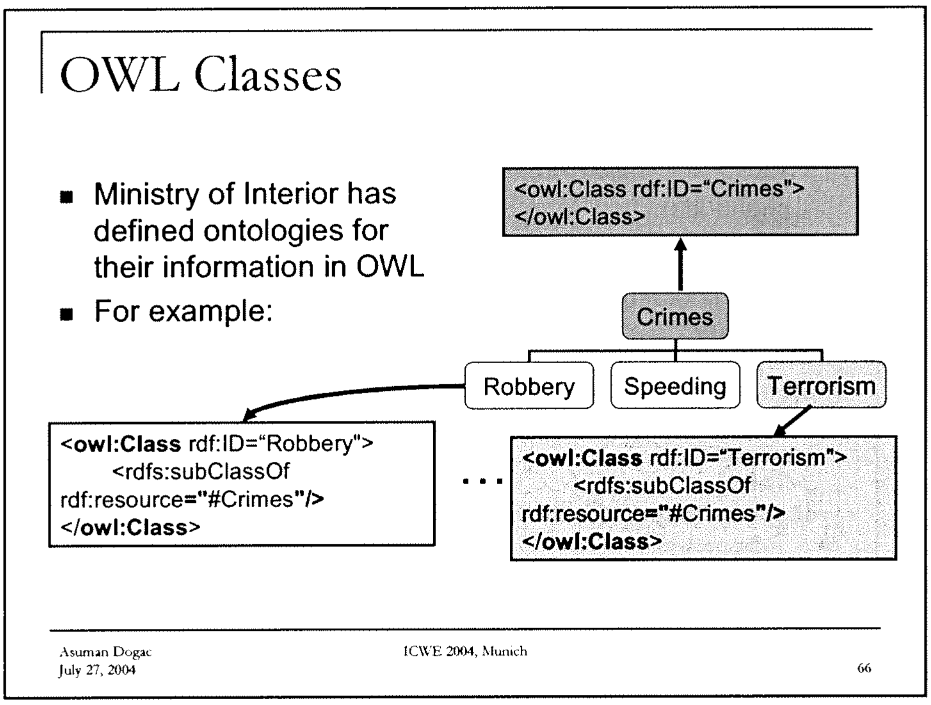
- **Class Axioms:**
  - *oneOf, dataRange*
  - *disjointWith*
  - *equivalentClass* (applied to class expressions)
  - *rdfs:subClassOf* (applied to class expressions)
- **Boolean Combinations of Class Expressions:**
  - *unionOf*
  - *intersectionOf*
  - *complementOf*
- **Arbitrary Cardinality:**
  - *minCardinality*
  - *maxCardinality*
  - *cardinality*
- **Filler Information:**
  - *hasValue*



An Example to Restriction

Example:

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



OWL Classes

- Ministry of Interior has defined ontologies for their information in OWL
- For example:

```
<owl:Class rdf:ID="Crimes">
</owl:Class>
```

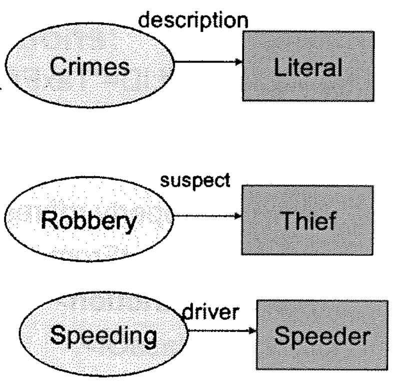
```
<owl:Class rdf:ID="Robbery">
      <rdfs:subClassOf
rdf:resource="#Crimes"/>
</owl:Class>
```

. . .

```
<owl:Class rdf:ID="Terrorism">
      <rdfs:subClassOf
rdf:resource="#Crimes"/>
</owl:Class>
```

OWL Properties

```
<owl:DatatypeProperty rdf:ID="description">
    <rdfs:domain rdf:resource="#Crime"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XML
    Schema#Literal"/>
</owl:DatatypeProperty >

<owl:ObjectProperty rdf:ID="suspect">
    <rdfs:domain rdf:resource="#Robbery"/>
    <rdfs:range rdf:resource="#Thief>
</owl:ObjectProperty >

<owl:ObjectProperty rdf:ID="driver">
    <rdfs:domain rdf:resource="#Speeding"/>
    <rdfs:range rdf:resource="#Speeder"/>
</owl:ObjectProperty >
```



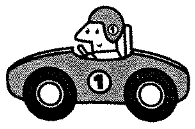
An Example (From Ref 3)

- Finger prints from a robbery scene identified John Smith as the suspect
- Here is the police report on the robbery:

```
<Robbery rdf:ID="report-2003-10-23-xyz">
   <description>...</description>
   <suspect>
   <Thief
rdf:about="http://www.ministryOfInterior.gov/criminals#John_Smith"/
>     </suspect>
</Robbery>
```

An Example (Continued)

- Later in the day a police gives a person a ticket for speeding
- The driver's license showed the name John Doe
- Here is the police report on the speeder:



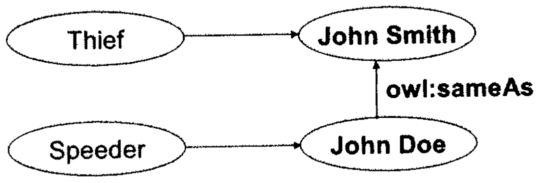
```
<Speeding rdf:ID="report-2003-10-23-abc">
   <description>...</description>
   <driver>
      <Speeder
rdf:about="http://www.ministryOfInterior/criminals#John_Doe"/>
   </driver>
</Speeding>
```

Any Relationship between the Thief and the Speeder?

Ministry of Interior keeps the OWL descriptions of their files:

```
<Criminals rdf:about="
        http://www.ministryOfInterior/criminals#John_Doe ">
  <owl:sameAs rdf:resource="
        http://www.ministryOfInterior.gov/criminals#John_Smith "/>
</Criminals>
```

An Example OWL Reasoning (Continued)

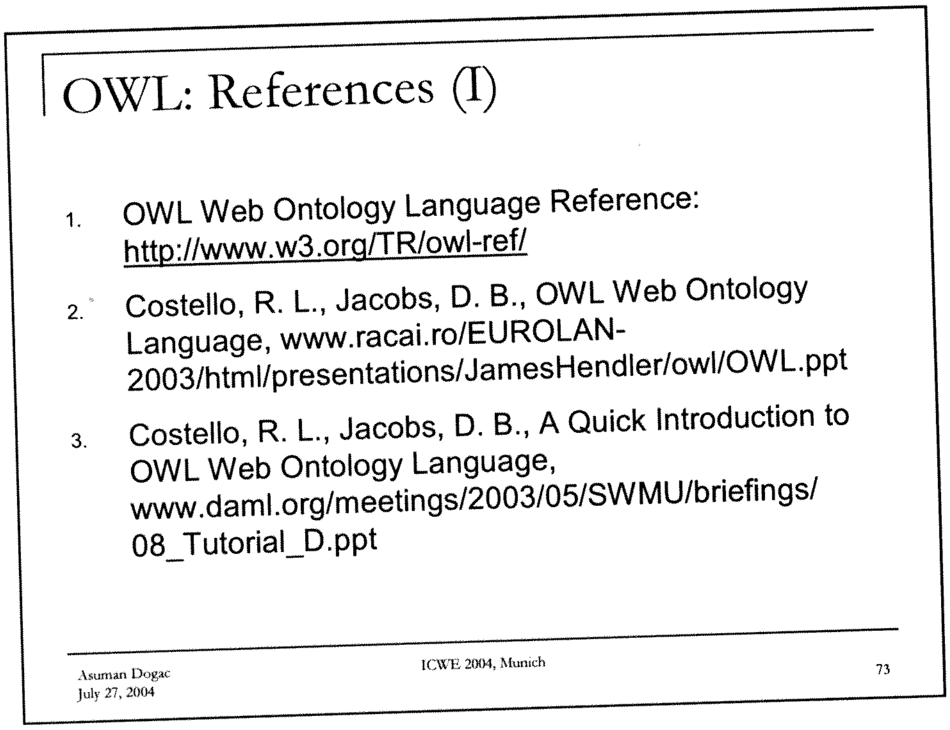


Inference: The Thief and the Speeder are one and the same!

- OWL provides a property (owl:sameAs) for indicating that two resources (e.g., two people) are the same

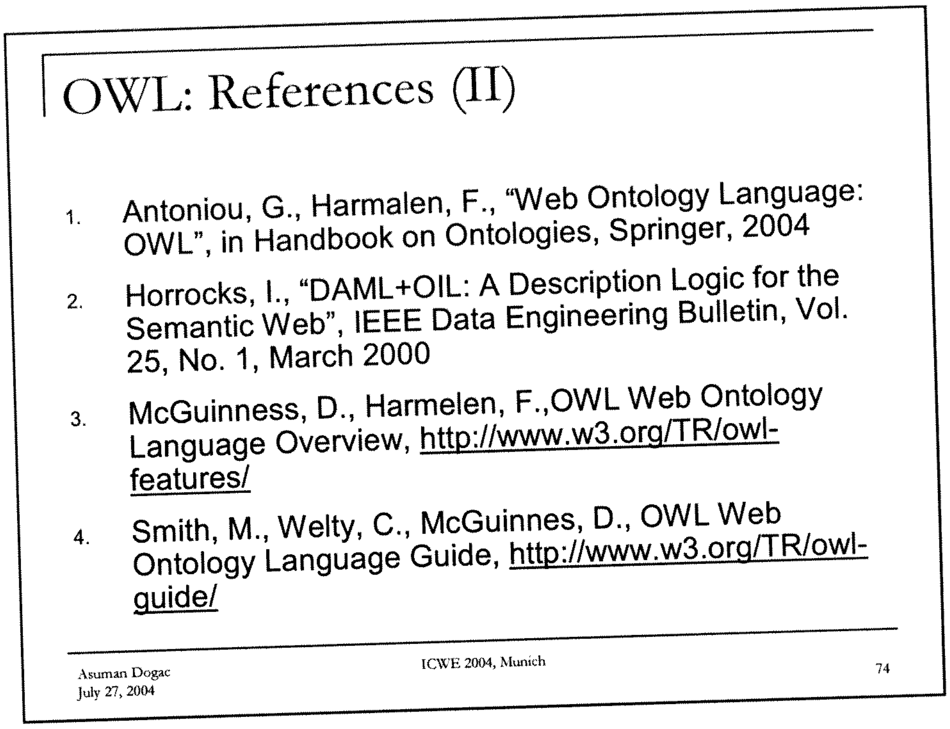
OWL Summary

- OWL is an Ontology Specification Language
- It is build on RDF with DAML+OIL experience
- OWL has more expressive power than RDF such as:
  - Boolean Combinations of Class Expressions (unionOf, intersectionOf, complementOf, ...)
  - Several Types of Properties (Transitive, Functional, Symmetric, ...)
  - Equivalence or disjointness of classes



OWL: References (I)

1. OWL Web Ontology Language Reference: http://www.w3.org/TR/owl-ref/
2. Costello, R. L., Jacobs, D. B., OWL Web Ontology Language, www.racai.ro/EUROLAN-2003/html/presentations/JamesHendler/owl/OWL.ppt
3. Costello, R. L., Jacobs, D. B., A Quick Introduction to OWL Web Ontology Language, www.daml.org/meetings/2003/05/SWMU/briefings/08_Tutorial_D.ppt



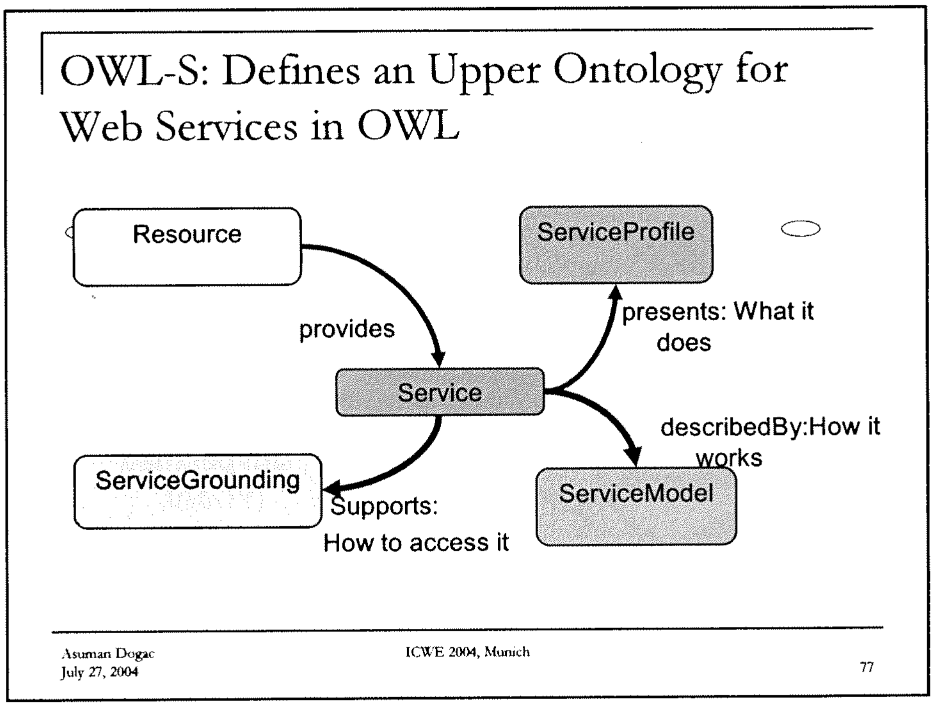
OWL: References (II)

1. Antoniou, G., Harmalen, F., "Web Ontology Language: OWL", in Handbook on Ontologies, Springer, 2004
2. Horrocks, I., "DAML+OIL: A Description Logic for the Semantic Web", IEEE Data Engineering Bulletin, Vol. 25, No. 1, March 2000
3. McGuinness, D., Harmelen, F.,OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/
4. Smith, M., Welty, C., McGuinnes, D., OWL Web Ontology Language Guide, http://www.w3.org/TR/owl-guide/

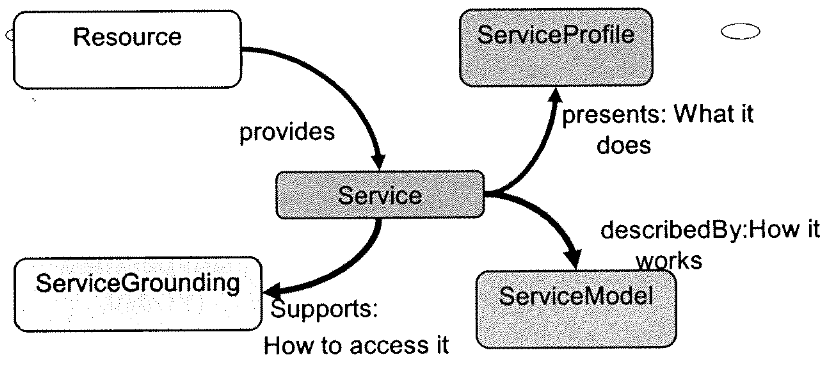
Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
  - **Ontology**
  - **RDF**
  - **OWL**
  - **OWL-S**
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions

OWL-S



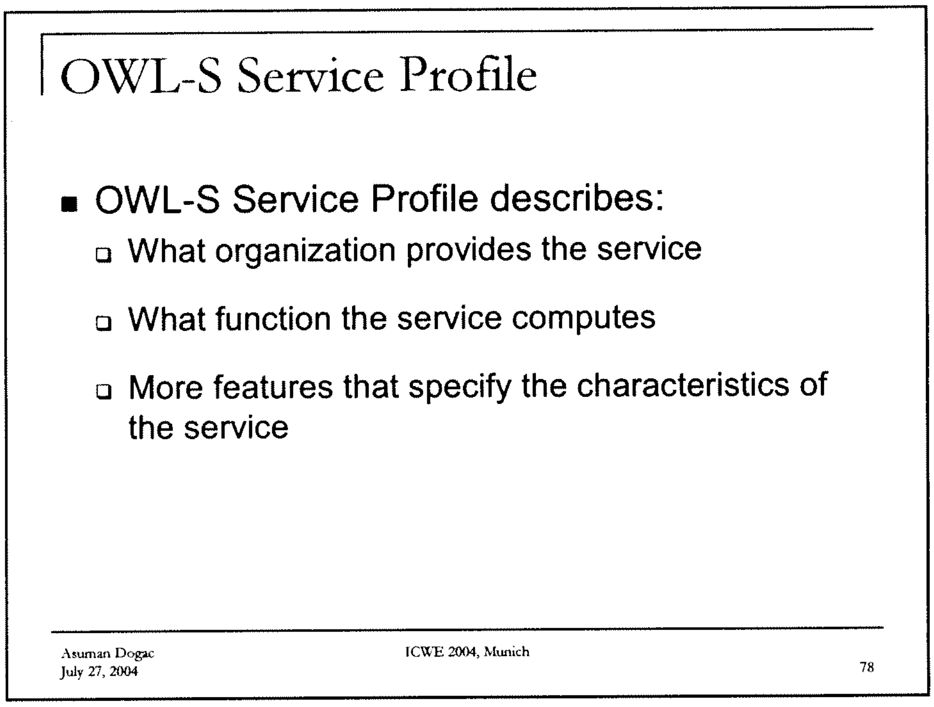
OWL-S: Defines an Upper Ontology for Web Services in OWL



Asuman Dogac
July 27, 2004

ICWE 2004, Munich

77



OWL-S Service Profile

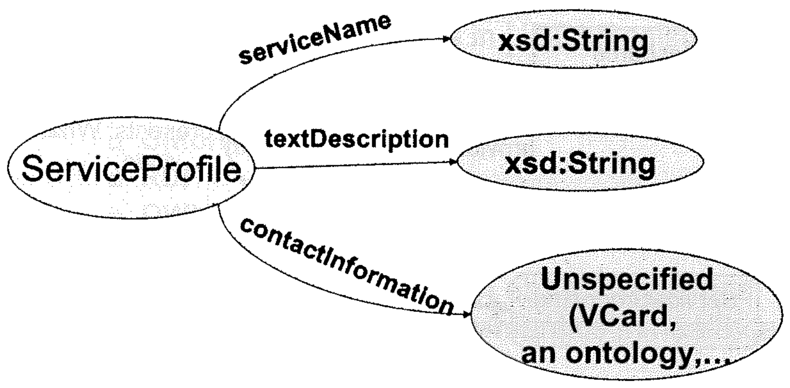
- OWL-S Service Profile describes:
  - What organization provides the service
  - What function the service computes
  - More features that specify the characteristics of the service

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

78

OWL-S Service Profile – Human Readable Info



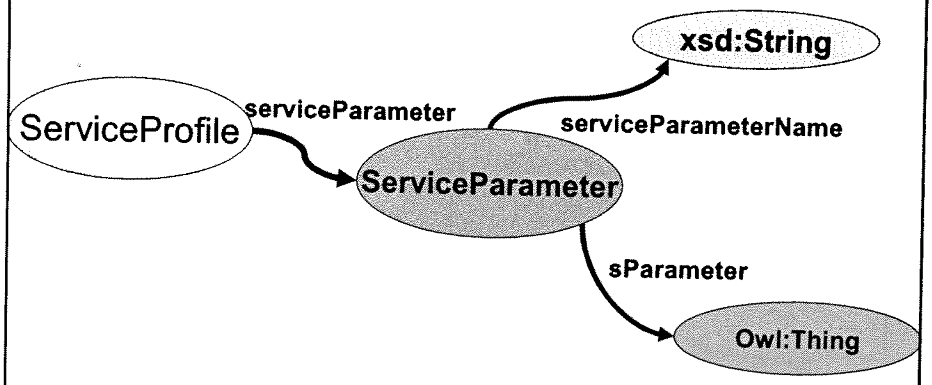
Example – Human Readable Info

<profile:serviceName>
Tax Heaven Tax Preparation Software
</profile:serviceName>

<profile:textDescription>
This service provides the tax form when appropriate income information is provided providing the best possible benefits to the tax payer ☺!
</profile:textDescription>

<profile:contactInformation> Vcard of the company
</profile:contactInformation>

OWL-S Service Profile – Service Parameter

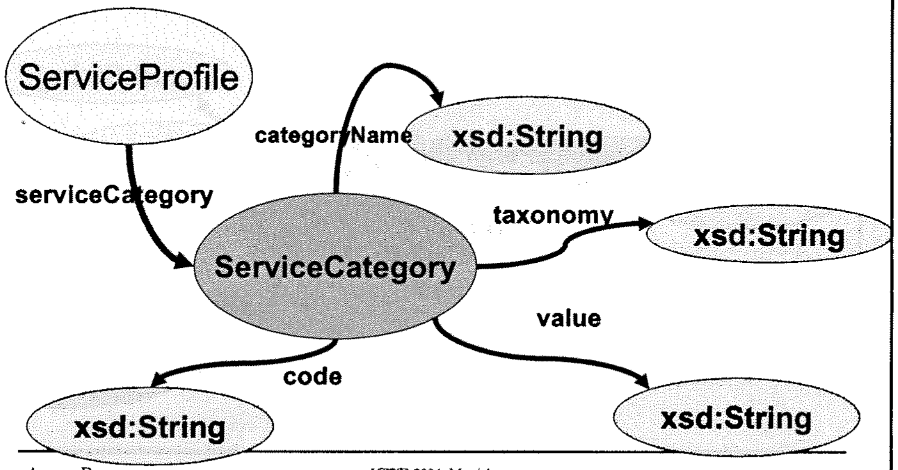


An Example – Service Parameter

```
<profile:serviceParameter>
    <addParam:GeographicRadius
       rdf:ID="TaxHeaven-geographicRadius">
       <profile:serviceParameterName>
              Tax Heaven Geographic Radius
       </profile:serviceParameterName>
       <profile:sParameter
              rdf:resource="&country;#UnitedStates"/>
    </addParam:GeographicRadius>
</profile:serviceParameter>
```

Note:xmlns:addParam="http://www.daml.org/services/owl-s/1.0/ProfileAdditionalParameters.owl#"

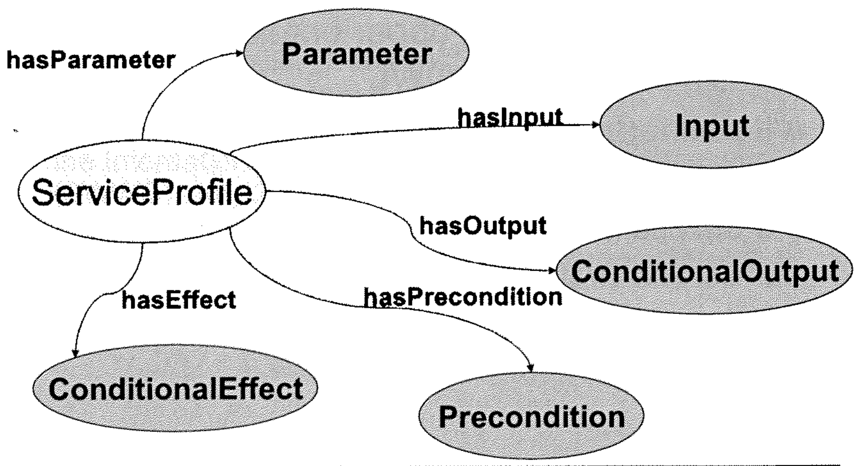
OWL-S Service Profile – Service Category



Example – Service Category

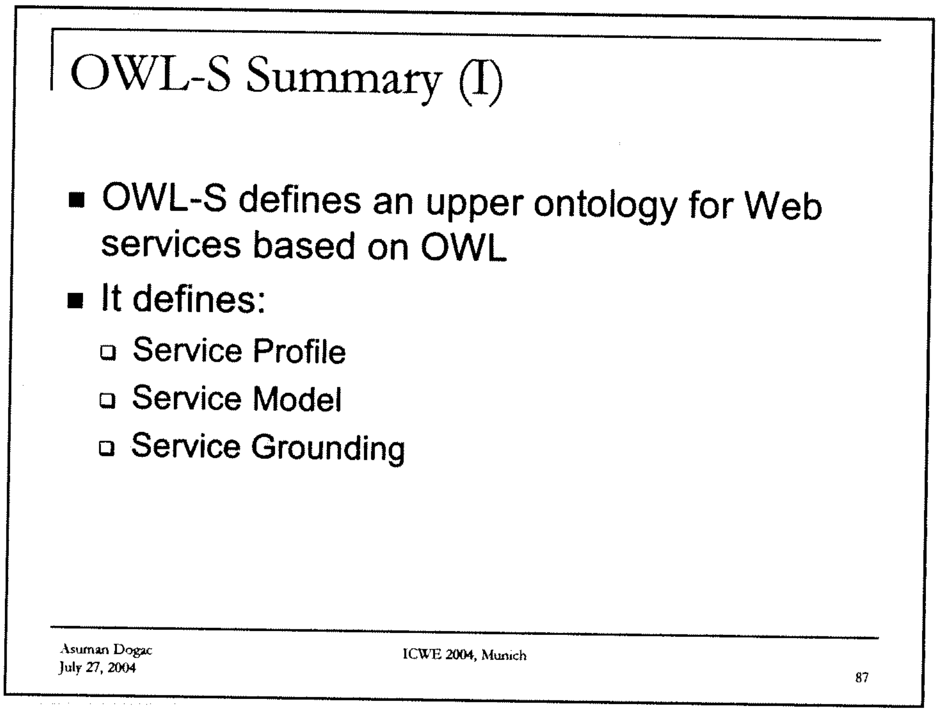
```
<profile:serviceCategory>
   <addParam:UNSPSC rdf:ID="UNSPSC-
                               category">
      <profile:value> Tax Preparation Software
      </profile:value>
      <profile:code> 43.16.17.02 </profile:code>
   </addParam:UNSPSC>
</profile:serviceCategory>
```

OWL-S Service Profile -IOPE



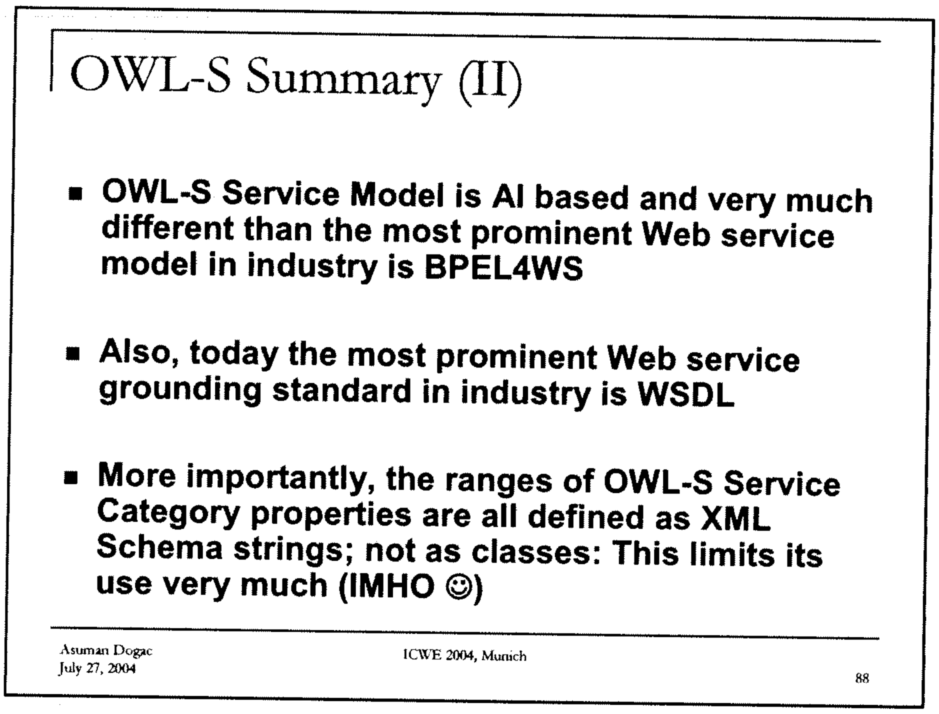
Example -IOPE

```
<profile:hasInput
  rdf:resource="&th_process;#TaxNumber"/>

<profile:hasInput
  rdf:resource="&th_process;#GrossIncome"/>

<profile:hasInput rdf:resource=
        "&th_process;#DeductableExpenses"/>

<profile:hasOutput rdf:resource=
        "&th_process;#TaxForm"/>
<profile:hasEffect rdf:resource=
  "&th_process;#ChargeForService"/>
```



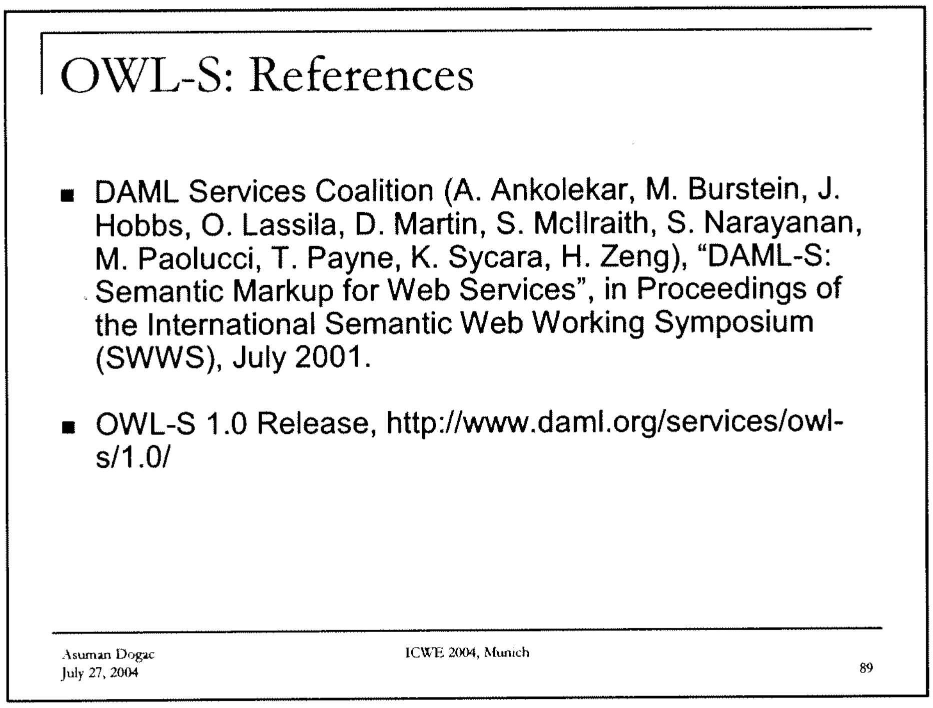
OWL-S Summary (I)

- OWL-S defines an upper ontology for Web services based on OWL
- It defines:
  - Service Profile
  - Service Model
  - Service Grounding



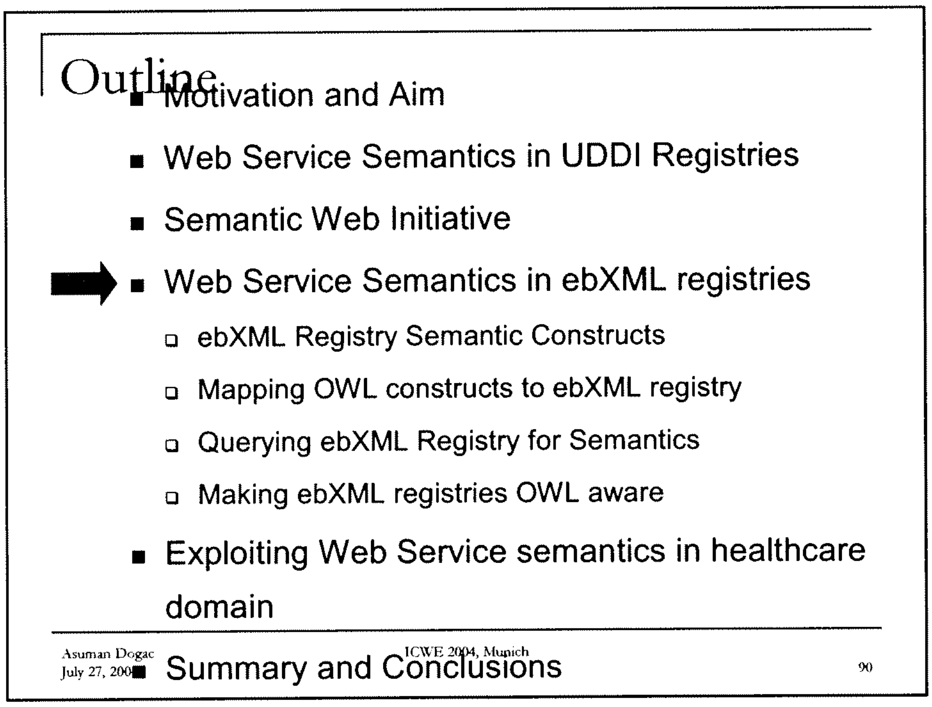
OWL-S Summary (II)

- **OWL-S Service Model is AI based and very much different than the most prominent Web service model in industry is BPEL4WS**
- **Also, today the most prominent Web service grounding standard in industry is WSDL**
- **More importantly, the ranges of OWL-S Service Category properties are all defined as XML Schema strings; not as classes: This limits its use very much (IMHO ☺)**



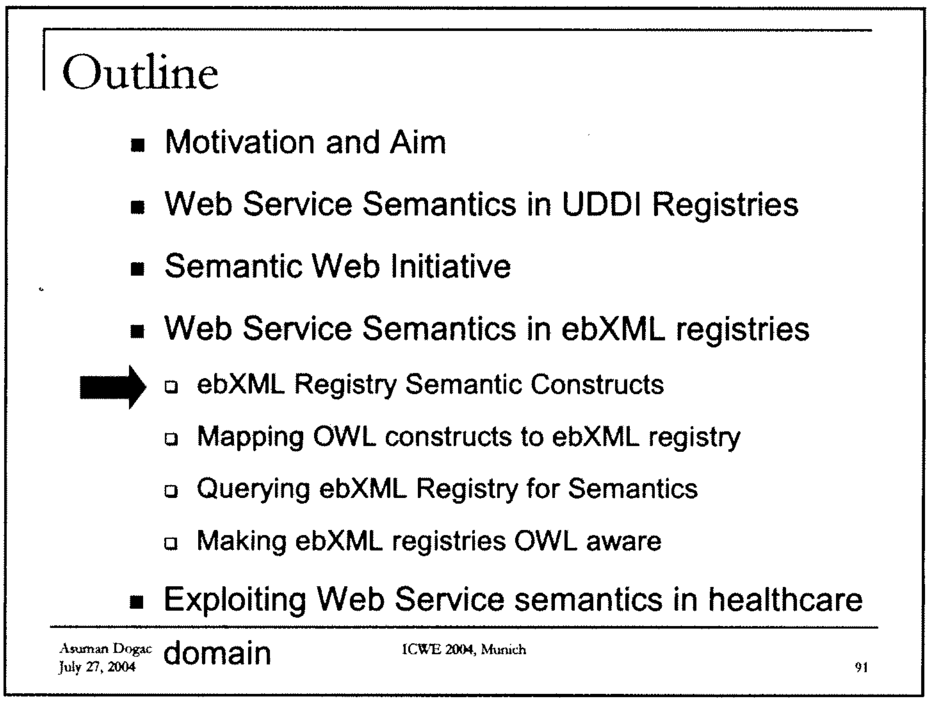
OWL-S: References

- 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.
- OWL-S 1.0 Release, http://www.daml.org/services/owl-s/1.0/



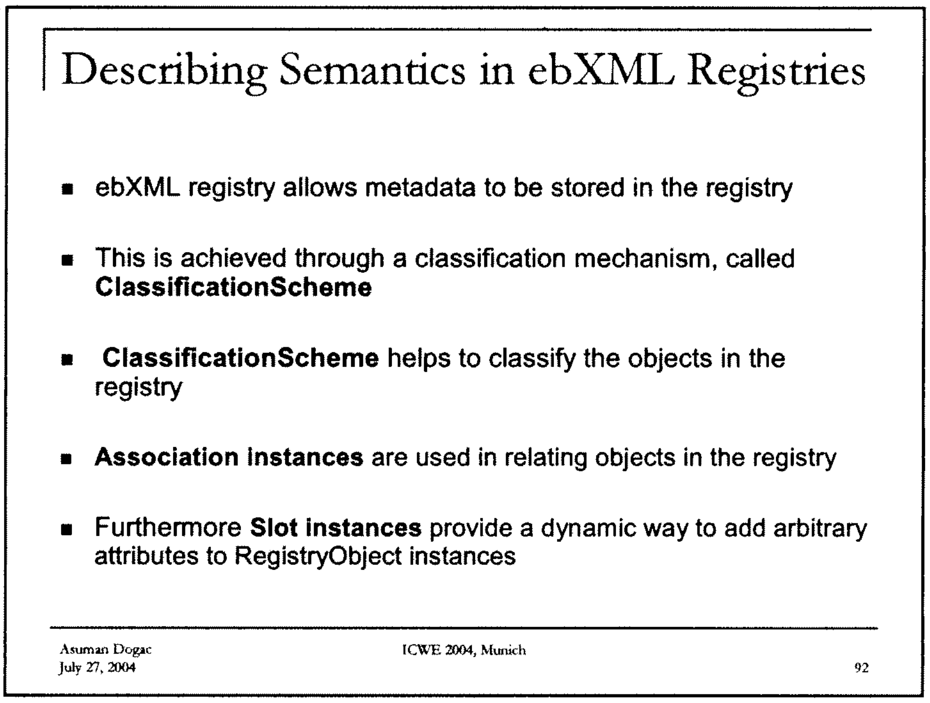
Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- ➡ Web Service Semantics in ebXML registries
  - ebXML Registry Semantic Constructs
  - Mapping OWL constructs to ebXML registry
  - Querying ebXML Registry for Semantics
  - Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain
- Summary and Conclusions



Outline

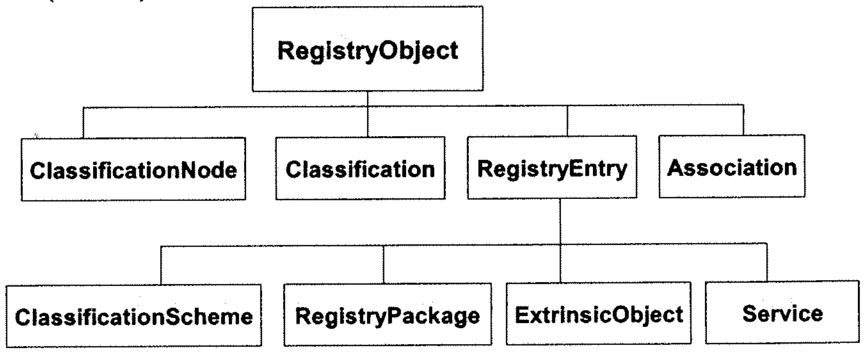
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
  - ➡ ebXML Registry Semantic Constructs
  - Mapping OWL constructs to ebXML registry
  - Querying ebXML Registry for Semantics
  - Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain



Describing Semantics in ebXML Registries

- ebXML registry allows metadata to be stored in the registry
- This is achieved through a classification mechanism, called **ClassificationScheme**
- **ClassificationScheme** helps to classify the objects in the registry
- **Association instances** are used in relating objects in the registry
- Furthermore **Slot instances** provide a dynamic way to add arbitrary attributes to RegistryObject instances

ebXML Registry Information Model (RIM)



ebXML Registry Information Model (RIM)

- **The RegistryObject class** is an abstract base class used by most classes in the model
- **Slot instances** provide a dynamic way to add arbitrary attributes to RegistryObject instances
- **Association instances** are RegistryObject instances that are used to define many- to-many associations between objects in the information model
- **ClassificationScheme** instances are RegistryEntry instances that describe a structured way to classify or categorize RegistryObject instances
- **ClassificationNode** instances are RegistryObject instances that are used to define tree structures under a ClassificationScheme
- **Classification instances** are RegistryObject instances that are used to classify other RegistryObject instances
- **RegistryPackage** instances are RegistryEntry instances that group logically related RegistryObject instances together

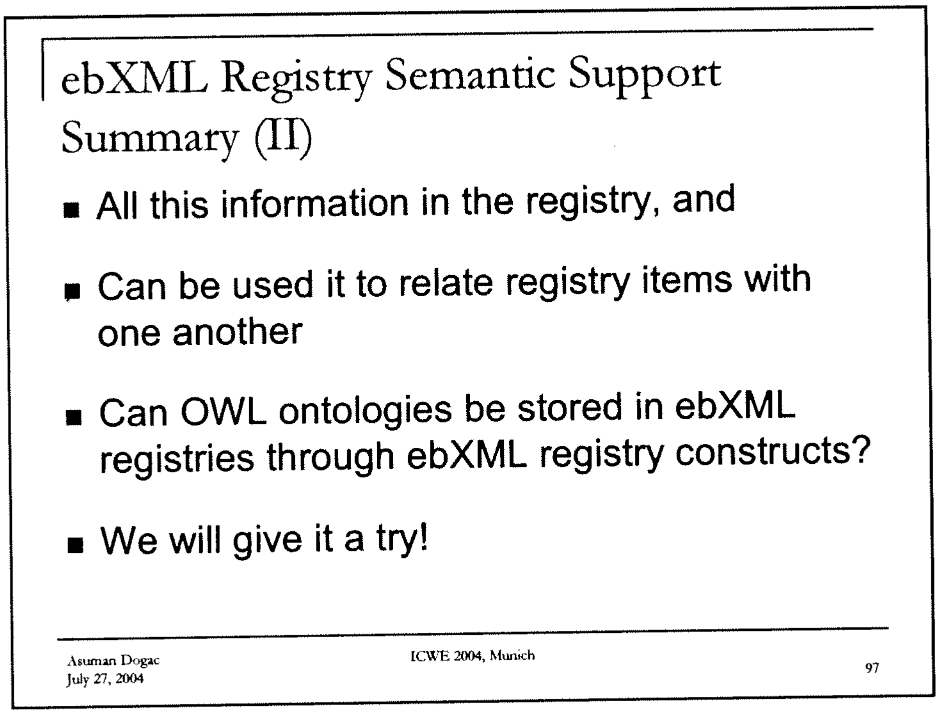
Some of the Predefined Association Types in ebXML Registries

RelatedTo	Relates Registry Objects
HasMember	Defines the members of the Registry Package
Contains	Defines that Source Registry Object contains the Target Registry Object
EquivalentTo	Defines that Source Registry Object is equivalent to the Target Registry Object
Extends	Defines that Source Registry Object inherits from the Target Registry Object
Implements	Defines that Source Registry Object implements the functionality defined by the Target Registry Object
InstanceOf	Defines that Source Registry Object is an instance of the Target Registry Object

ebXML allows this list to be extended!

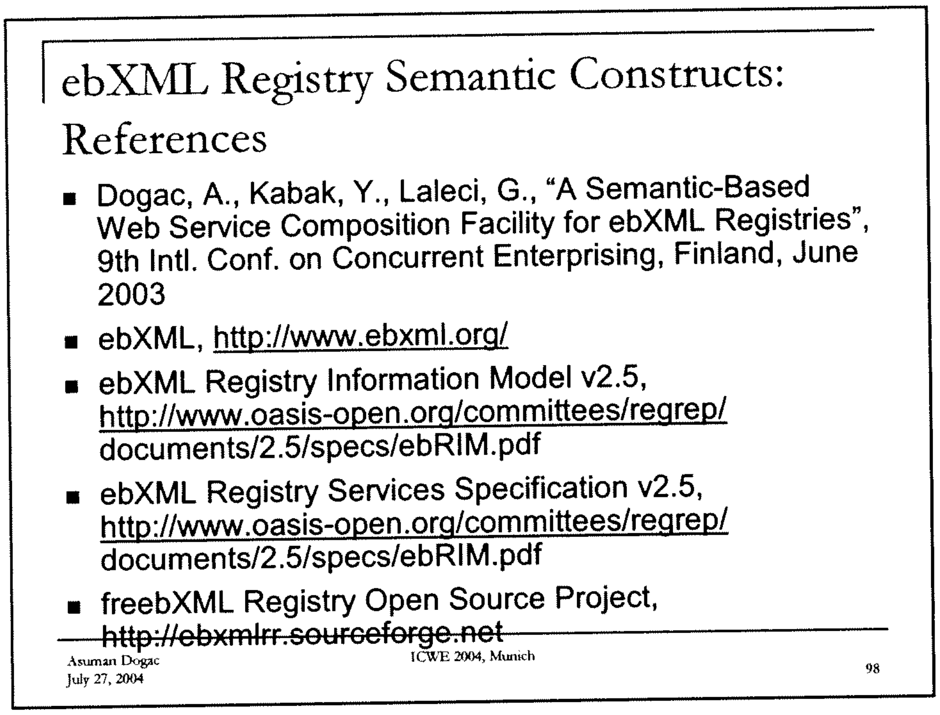
ebXML Registry Semantic Support Summary (I)

- Through the constructs provided in an ebXML registry, it is possible to define:
  - Classes and class hierarchies
  - Properties through “slot” mechanism
  - Properties of classes through predefined association types
  - Predefined associations can be extended
  - Group registry objects



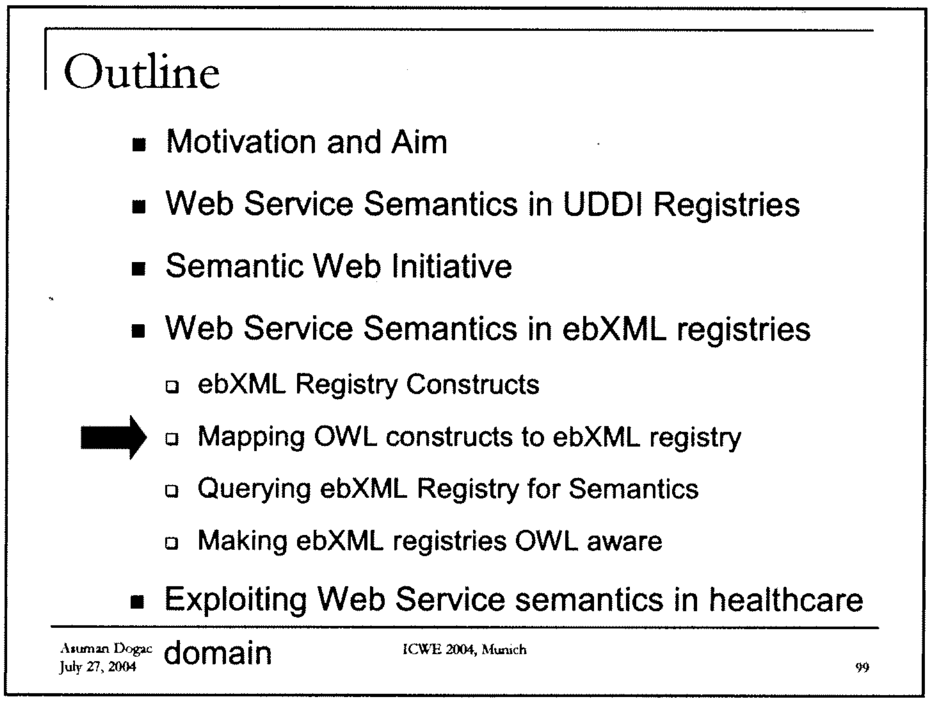
ebXML Registry Semantic Support Summary (II)

- All this information in the registry, and
- Can be used it to relate registry items with one another
- Can OWL ontologies be stored in ebXML registries through ebXML registry constructs?
- We will give it a try!



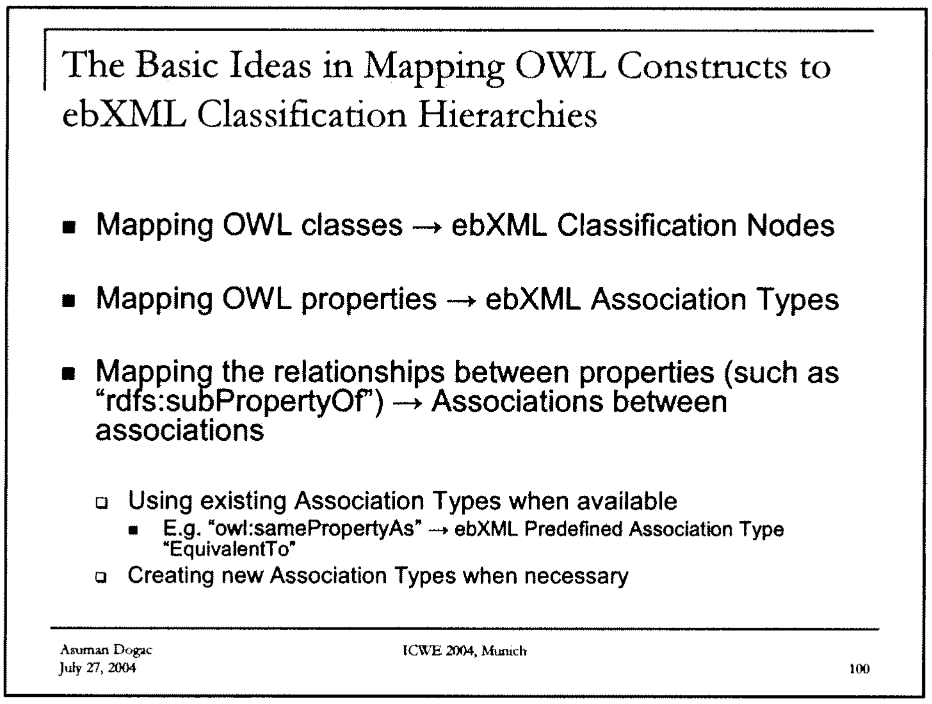
ebXML Registry Semantic Constructs: References

- Dogac, A., Kabak, Y., Laleci, G., "A Semantic-Based Web Service Composition Facility for ebXML Registries", 9th Intl. Conf. on Concurrent Enterprising, Finland, June 2003
- ebXML, http://www.ebxml.org/
- ebXML Registry Information Model v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebRIM.pdf
- ebXML Registry Services Specification v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebRIM.pdf
- freebXML Registry Open Source Project, http://ebxmlrr.sourceforge.net



Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
  - ebXML Registry Constructs
  - Mapping OWL constructs to ebXML registry
  - Querying ebXML Registry for Semantics
  - Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain



The Basic Ideas in Mapping OWL Constructs to ebXML Classification Hierarchies

- Mapping OWL classes → ebXML Classification Nodes
- Mapping OWL properties → ebXML Association Types
- Mapping the relationships between properties (such as "rdfs:subPropertyOf") → Associations between associations
  - Using existing Association Types when available
    - E.g. "owl:samePropertyAs" → ebXML Predefined Association Type "EquivalentTo"
  - Creating new Association Types when necessary

The Basic Ideas in Mapping OWL Constructs to ebXML Classification Hierarchies

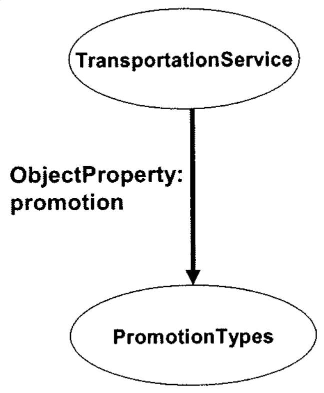
- Mapping OWL collections → ebXML RegistryPackage
- How to handle OWL multiple inheritance?
- How to handle OWL Class Boolean expressions?
- How to handle OWL restriction?

OWL Classes and RDF Properties → ebXML RIM

OWL classes → ebXML ClassificationNodes
OWL properties → ebXML new Association Types
Example:

OWL:

```
<owl:Class rdf:ID="TransportationService">
<rdfs:subClassOf rdf:resource="#TravelService"/>
</owl:Class>

<owl:Class rdf:ID="PromotionTypes">
<rdfs:subClassOf rdf:resource="#TravelService"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="promotion">
  <rdfs:domain rdf:resource="#TransportationService"/>
  <rdfs:range rdf:resource="#PromotionTypes"/>
</owl:ObjectProperty>
```

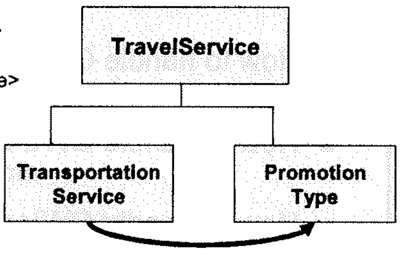


OWL Classes and RDF Properties → ebXML RIM

- **OWL classes → ClassificationNodes**
- **OWL Properties → ebXML new Associations Types**
- **Example:**

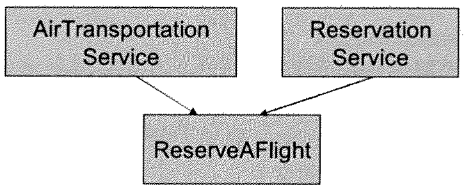
ebXML RIM:

```
<rim:ClassificationNode id =
'TransportationService' parent= 'TravelService'>
   <rim:Name> <rim:LocalizedString value =
          "TransportationService"/> </rim:Name>
</rim:ClassificationNode>

<rim:ClassificationNode id = 'PromotionType'
parent= 'TravelService'>
<rim:Name> <rim:LocalizedString value =
          "PromotionType"/> </rim:Name>
</rim:ClassificationNode>

<rim:Association id = 'promotion' associationType =
'ObjectProperty'
sourceObject = 'TransportationService'
targetObject = 'PromotionType' >
<rim:Name> <rim:LocalizedString value =
          "promotion"/> </rim:Name>
</rim:Association>
```



Association of type "ObjectProperty": promotion

OWL Multiple Inheritance → ebXML RIM New Association Type: subClassOf



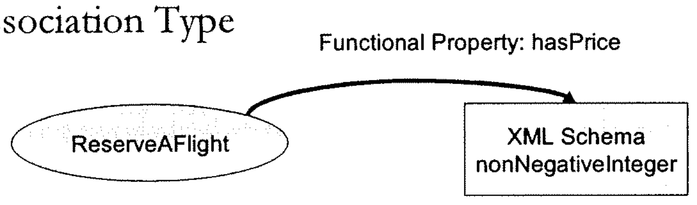
OWL:

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

ebXML:

```
<rim:Association id = 'RAsubclass'
associationType = 'subClassOf '
sourceObject = 'ReserveAFlight'
targetObject =
`AirTransportationService'><rim:Name>
<rim:LocalizedString value =
"RAsubclass"/> </rim:Name>
</rim:Association>

<rim:Association id = 'RRsubclass '
associationType = 'subClassOf '
sourceObject = 'ReserveAFlight '
targetObject = `ReservationService'>
<rim:Name> <rim:LocalizedString value =
"RRSubclass"/> </rim:Name>
</rim:Association>
```

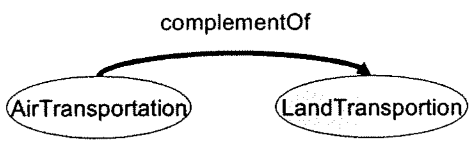
OWL Functional Property → ebXML RIM New Association Type



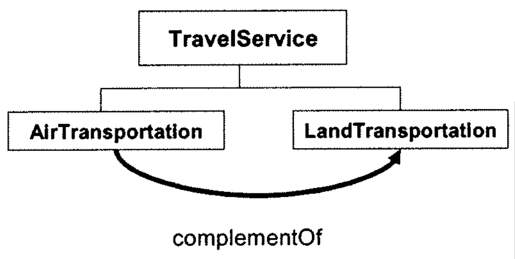
```
<owl:FunctionalProperty rdf:ID="hasPrice">
  <rdfs:subpropertyOf
    rdf:resource="http://www.daml.org/
       services/daml-s/2001/05/Profile.owl"/>
  <rdfs:domain
    rdf:resource="#ReserveAFlight"/>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/10/
    XMLSchema/nonNegativeInteger"/>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

```
<rim:Association id = 'hasPrice'
associationType = 'FunctionalProperty'
sourceObject = 'ReserveAFlight'
targetObject = 'XML Schema
nonNegativeIntegerDataType' >
<rim:Name> <rim:LocalizedString value =
            "hasPrice"/> </rim:Name>
</rim:Association>
```

OWL Boolean Combination of Classes → ebXML RIM New Association Type



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



```
<rim:Association id = 'ALComp'
associationType = 'complementOf'
sourceObject = 'AirTransportation '
targetObject = 'LandTransportion ' >
<rim:Name> <rim:LocalizedString value
=  "ALComp"/> </rim:Name>
</rim:Association>
```

OWL Restriction

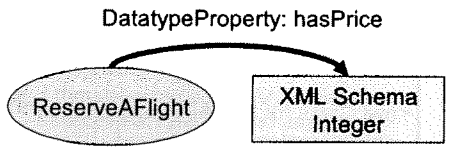
- In RDF, a property has a global scope, that is, no matter what class the property is applied to, the range of the property is the same
- owl:Resriction, on the other hand, has a local scope; restriction is applied on the property within the scope of the class where it is defined
- In ebXML class hierarchies, an association (which represents a property) is defined already in a local scope by associating two nodes of the class hierarchy

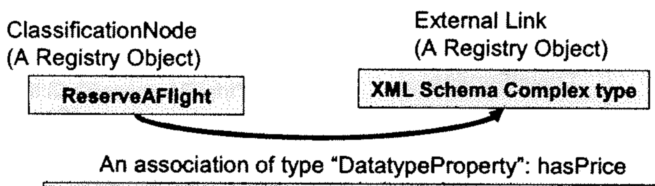
OWL provides the following language elements to indicate the type of restriction:
• owl:allValuesFrom
• owl:someValuesFrom
• owl:hasValue

Example:
```
<owl:Class rdf:ID="ReserveAFlight">
<rdfs:subClassOf>
   <owl:Restriction>
   <owl:onProperty rdf:resource="#paymentMethod"/>
   <owl:allValuesFrom rdf:resource= "#CreditCard"/>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

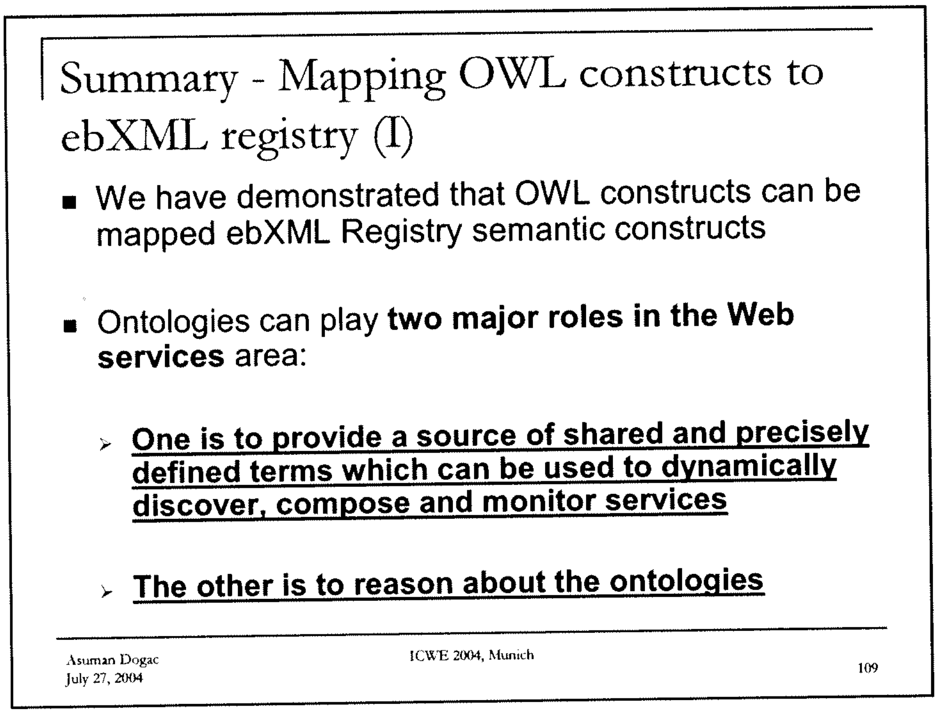
OWL XML Schema Datatypes

- 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, basic XML Schema datatypes are stored in ebXML registry
- For complex types an external link is provided from the registry



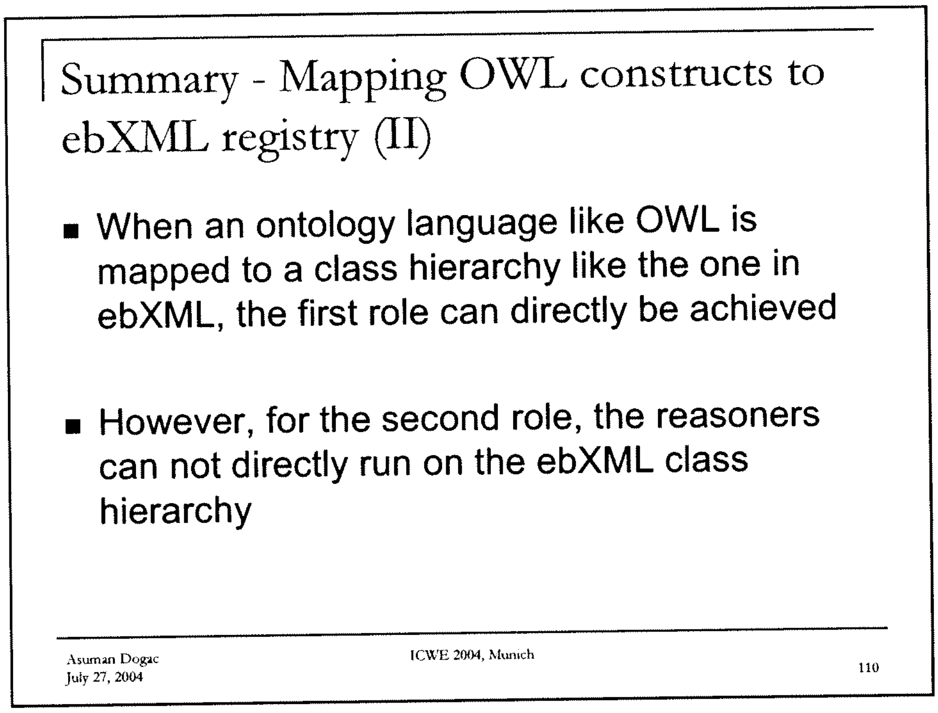


```
<rim:ExternalLink id = "hasPrice"
        externalURI="http://www.w3.org/2001/XMLSchema#complexType" >
  <rim:Name> <rim:LocalizedString value = "XML Schema Complex Type"/>
   </rim:Name>
</rim:ExternalLink>
```



Summary - Mapping OWL constructs to ebXML registry (I)

- We have demonstrated that OWL constructs can be mapped ebXML Registry semantic constructs
- 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**



Summary - Mapping OWL constructs to ebXML registry (II)

- 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 can not directly run on the ebXML class hierarchy

Summary - Mapping OWL constructs to ebXML registry (III)

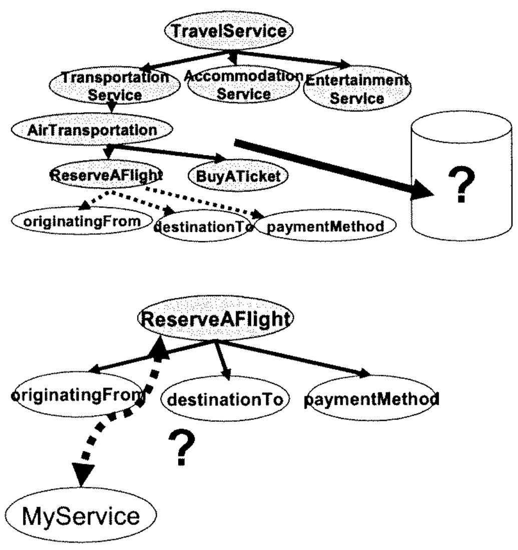
- Some OWL constructs are for reasoners to use them
- The fact is that we do not have industrial strength reasoners yet!
- Semantic can also be taken advantage of through querying
- Main reference: Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004

Outline

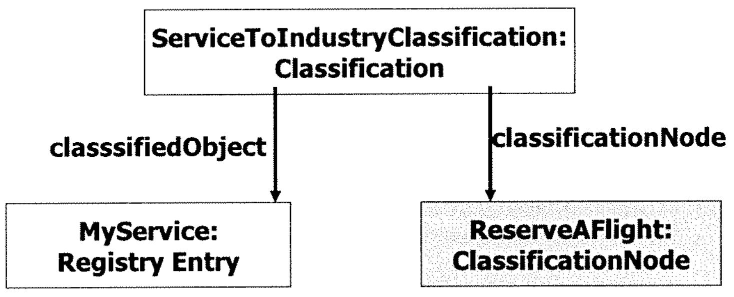
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
  - ebXML Registry Semantic Constructs
  - Mapping OWL constructs to ebXML registry
  - ➡ Querying ebXML Registry for Semantics
  - Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain

How to Relate Semantics with Registry Instances

- In relating the semantics with the services advertised in service registries, there are two key issues:
  - Where to store the generic semantics of the services: In ebXML, metadata is stored in the registry
  - How to relate the services advertised in the registry with the semantic defined through an ontology: In ebXML through Classification objects



Relating a Web service Advertised with Service Ontology in ebXML


ServiceToIndustryClassification: Classification
classsifiedObject
classificationNode
MyService: Registry Entry
ReserveAFlight: ClassificationNode

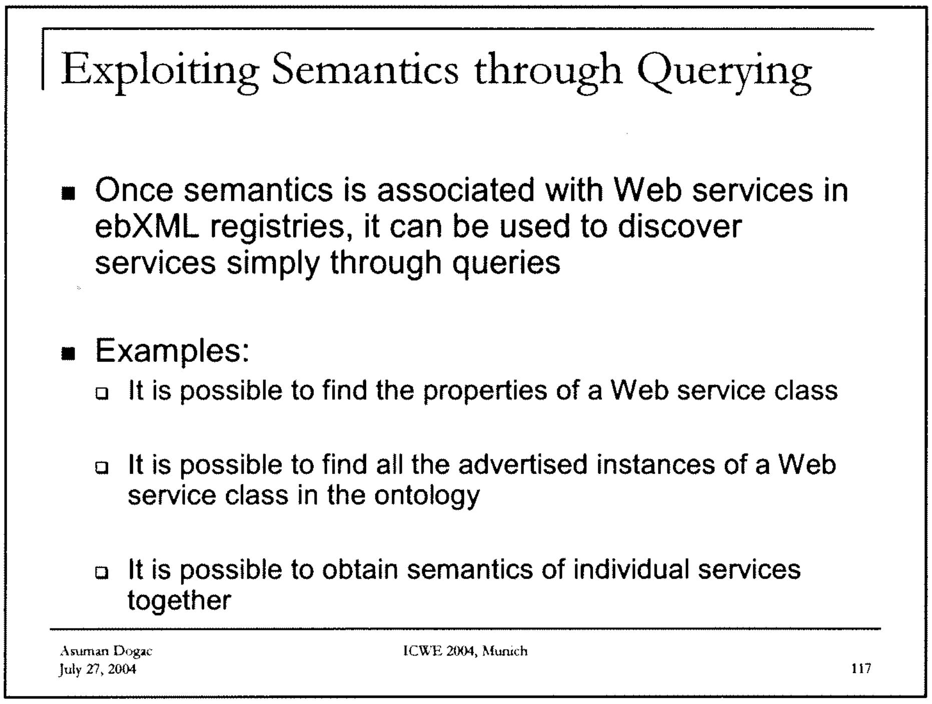
How to relate services advertised with the generic ontology classes?

- By relating a service advertised with a node in classification hierarchy, we make the service an explicit member of this node
- The service also inherits the well-defined meaning associated with this node as well as the generic properties defined for this node
- When we associate "MyService" with "ReserveAFlightService", its meaning becomes clear; that this service is a flight reservation service
- Assuming that the "ReserveAFlightService" service has the generic properties such as "originatingFrom", "destinationTo" and "paymentMethod", "MyService" also inherits these properties

How to relate services advertised with the generic ontology classes?

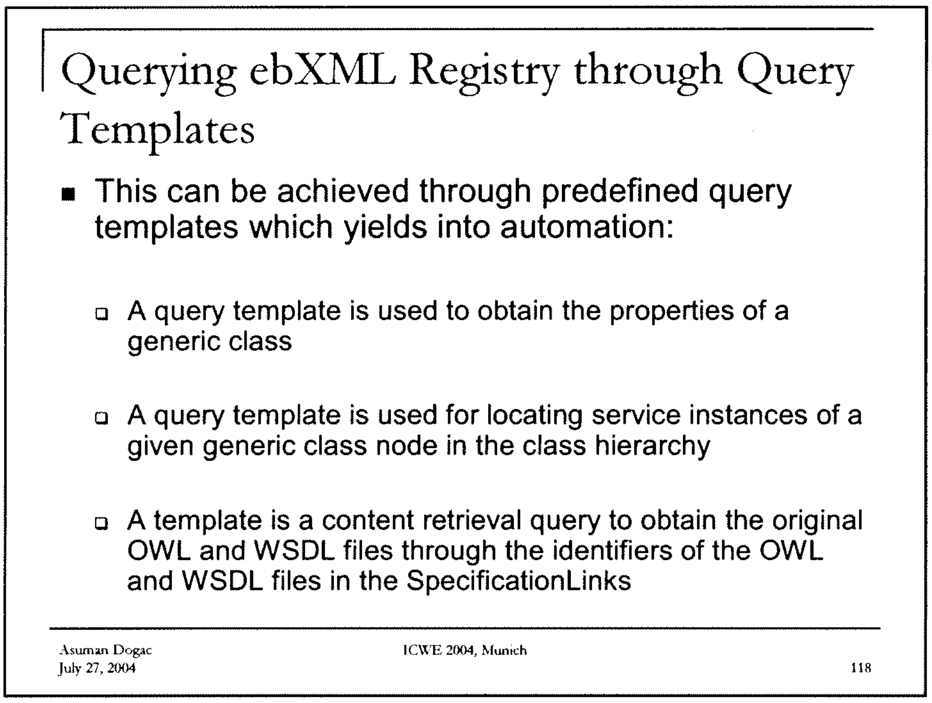
The following "SubmitObjectsRequest" classifies "MyReserveAFlightService" with "ReserveAFlight" ClassificationNode

```
<rs:SubmitObjectsRequest >
 <LeafRegistryObjectList>
  <Service id="MyReserveAFlightService">
   <Name>
    <LocalizedString lang="en_US" value ="
    MyReserveAFlightService "/>
   </Name> </Service>
   <Classification classificationNode="ReserveAFlight"
     classifiedObject="MyReserveAFlightService" />
  </LeafRegistryObjectList>
</rs:SubmitObjectsRequest>
```



Exploiting Semantics through Querying

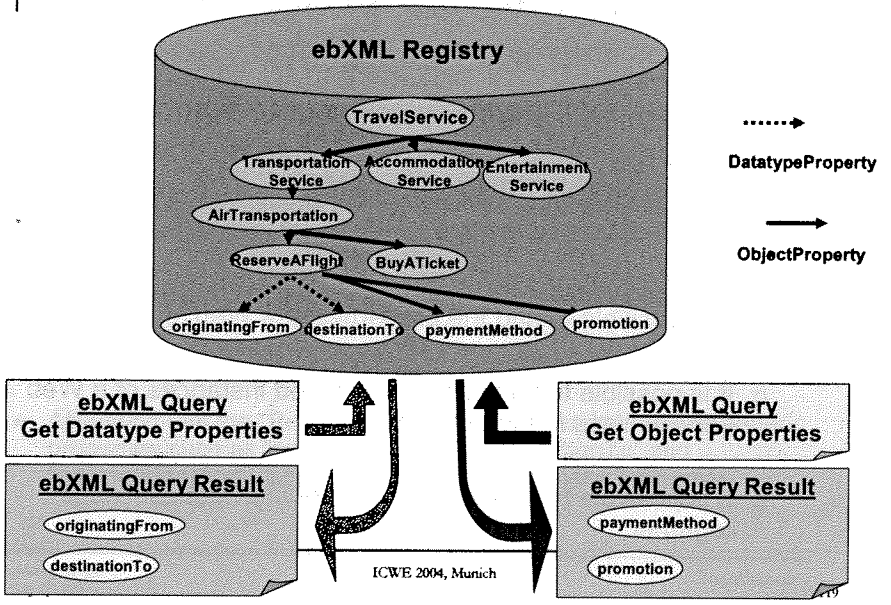
- Once semantics is associated with Web services in ebXML registries, it can be used to discover services simply through queries
- Examples:
  - It is possible to find the properties of a Web service class
  - It is possible to find all the advertised instances of a Web service class in the ontology
  - It is possible to obtain semantics of individual services together



Querying ebXML Registry through Query Templates

- This can be achieved through predefined query templates which yields into automation:
  - A query template is used to obtain the properties of a generic class
  - A query template is used for locating service instances of a given generic class node in the class hierarchy
  - A 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

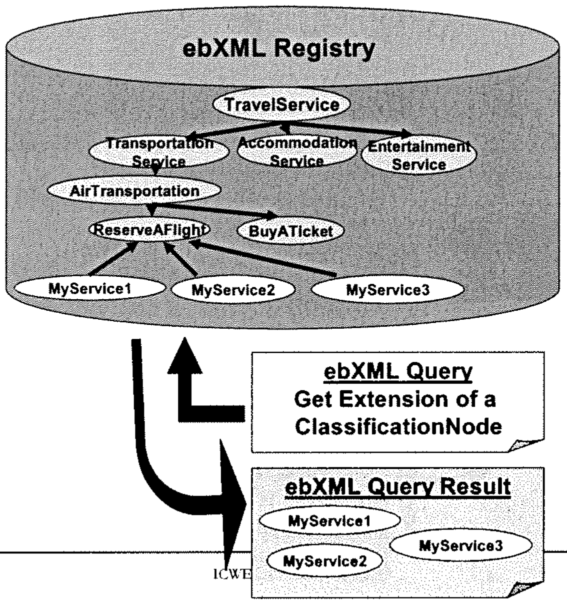
A query template to obtain the properties of a generic class



An Example Query Retrieving all the Associations of Type "dataTypeProperty" for "ReserveAClassFlightService"

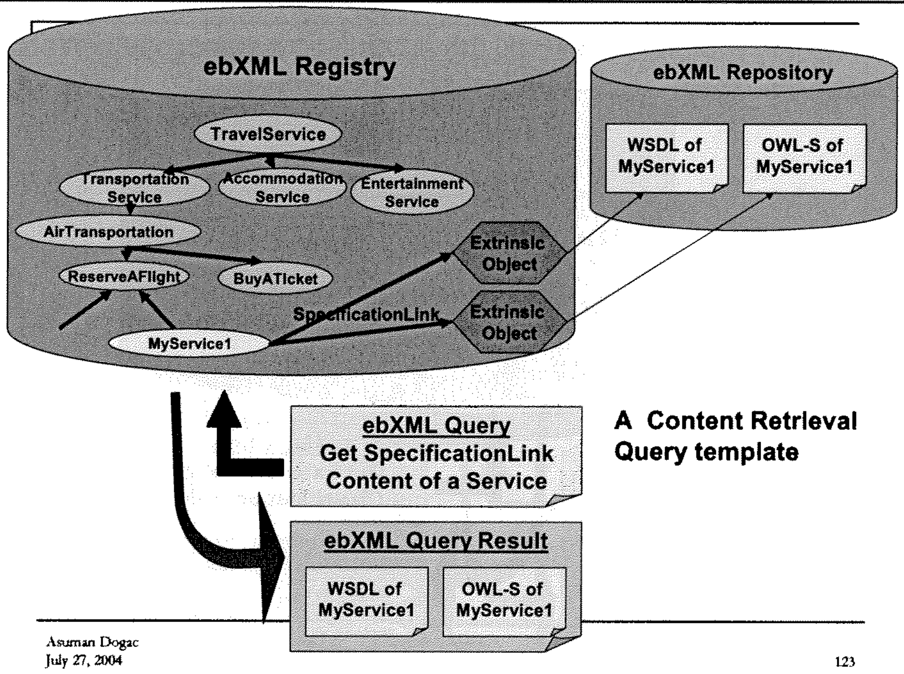
```
<AdhocQueryRequest xmlns =
"urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0" xmlns:xsi =
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation =
"urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0 query.xsd">
      <ResponseOption returnType = "LeafClass" returnComposedObjects = "true" />
      <FilterQuery>   <ClassificationNodeQuery>  <SourceAssociationBranch>
          <AssociationFilter>  <Clause>
                <SimpleClause leftArgument = "associationType">
                  <StringClause stringPredicate = "Equal">
                              dataTypeProperty</StringClause>
               </SimpleClause>  </Clause>
           </AssociationFilter>
        <ClassificationNodeQuery> <NameBranch>
            <LocalizedStringFilter> <Clause> <SimpleClause leftArgument = "value">
                       <StringClause stringPredicate = "Contains">
                                       ReserveAFlightService</StringClause>
                 </SimpleClause> </Clause> </LocalizedStringFilter>  </NameBranch>
      </ClassificationNodeQuery> </SourceAssociationBranch>
   </ClassificationNodeQuery>  </FilterQuery> </AdhocQueryRequest>
```

A query template to find all the advertised instances of a Web service class



An Example Query: Retrieving all the Services Classified with “ReserveAFlightService” ClassificationNode

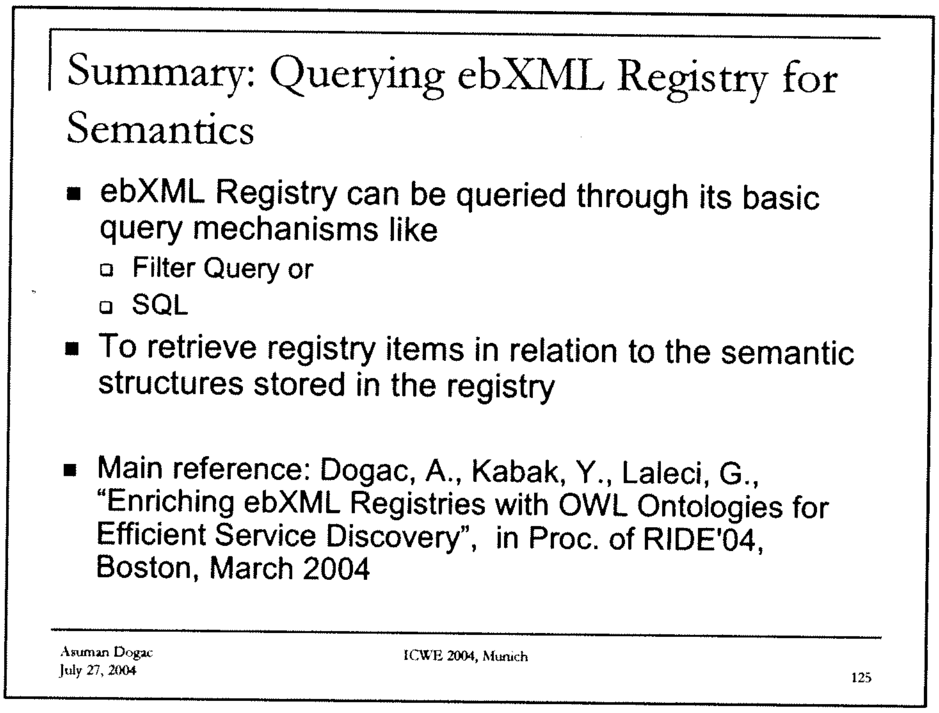
```
<AdhocQueryRequest
 xmlns = "urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation = "urn:oasis:names:tc:ebxml-regrep:query:xsd:  2.0 query.xsd">
 <ResponseOption returnType = "LeafClass" returnComposedObjects =  "true" />
  <FilterQuery>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



A Content Retrieval Query template

Retrieving the WSDL Files (Content Retrieval Query)

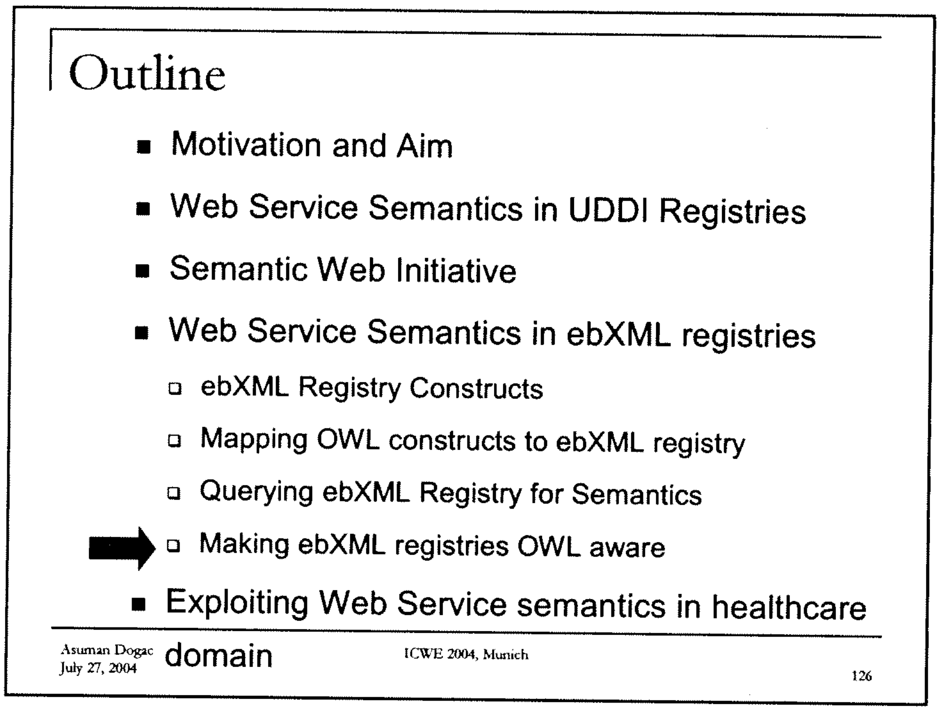
```
<GetContentRequest
    xmlns="urn:oasis:names:tc:ebxml-regrep:query:xsd:2.1"
    xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1
    ../schema/rim.xsd urn:oasis:names:tc:ebxml-regrep:query:xsd:2.1
    ../schema/query.xsd">
        <rim:ObjectRefList>
        <--! The unique id of the WSDL file in the registry -->
        <rim:ObjectRef
            id="urn:uuid:7e4397db-916a-490f-bdc7-c9da"/>
        </rim:ObjectRefList>
</GetContentRequest>
```



Summary: Querying ebXML Registry for Semantics

- ebXML Registry can be queried through its basic query mechanisms like
  - Filter Query or
  - SQL
- To retrieve registry items in relation to the semantic structures stored in the registry

- Main reference: Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004



Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
  - ebXML Registry Constructs
  - Mapping OWL constructs to ebXML registry
  - Querying ebXML Registry for Semantics
  - ➡ Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain

A Two Seconds Introduction to ebXML Registry Implementation

ebXML Registry Implementation: Relational Database with Stored Procedures

- ebXML registry implementations store registry data in a relational database
- The Query Manager component constructs the objects by obtaining the required data from the relational database through SQL queries
- When a client submits a request to the registry, registry objects are constructed by retrieving the related information from the database through SQL queries and are served to the user through the methods of these objects

Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
  - ebXML Registry Constructs
  - Mapping OWL constructs to ebXML registry
  - Querying ebXML Registry for Semantics
  - ➡ Making ebXML registries OWL aware
- Exploiting Web Service semantics in healthcare domain

Making ebXML registries OWL aware

- As already demonstrated OWL constructs can be mapped to ebXML registry information model constructs
- In this process, the ebXML Registry architecture is not modified
- In this way, the semantic explicitly stored in the registry can be retrieved through querying

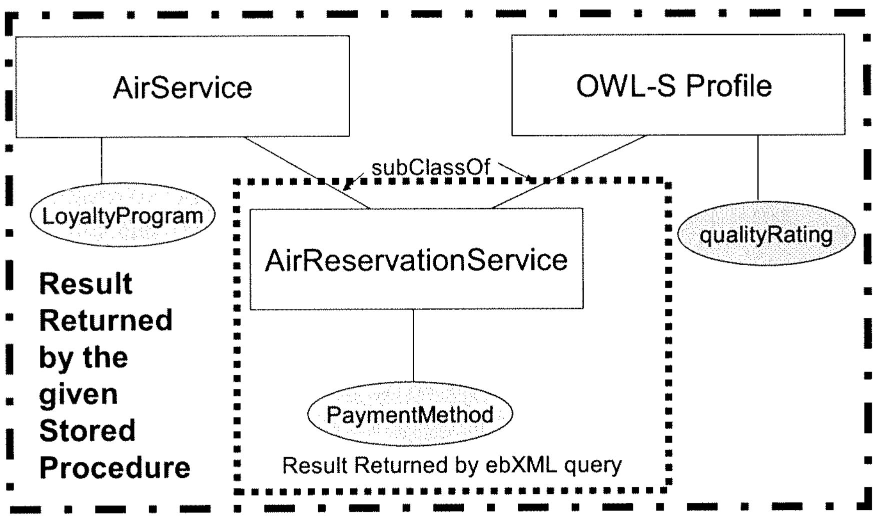
Making ebXML registries OWL aware

- For example, "subClassOf" association is defined in ebXML registry to express "multiple inheritance"
- Yet to make any use of this semantics, the user must code the query, say, to find out all the super classes of a given class
- An improvement: The code to process the OWL semantics can be stored in ebXML registry architecture through predefined procedures

Example: A Stored Procedure to Find Super Classes of a Given Class

```
CREATE PROCEDURE findSuperClasses($className)
   AS
BEGIN
SELECT C2.id
FROM Association A, Name_ N, ClassificationNode C1,
   ClassificationNode C2
WHERE A.associationType LIKE 'subClassOf' AND
    C1.id = N.parent AND
    N.value LIKE $className AND
    A.sourceObject = C1.id AND
    A.targetObject = C2.id
END;
```

An Example



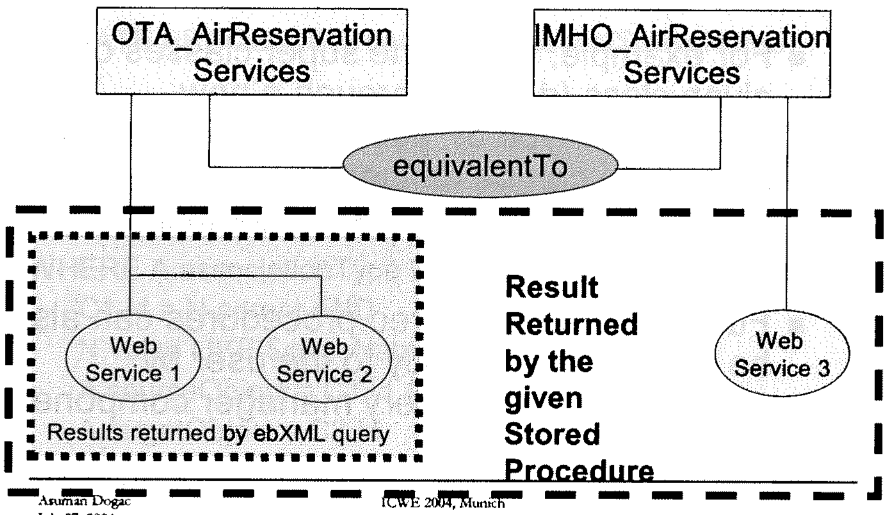
Example (Continued)

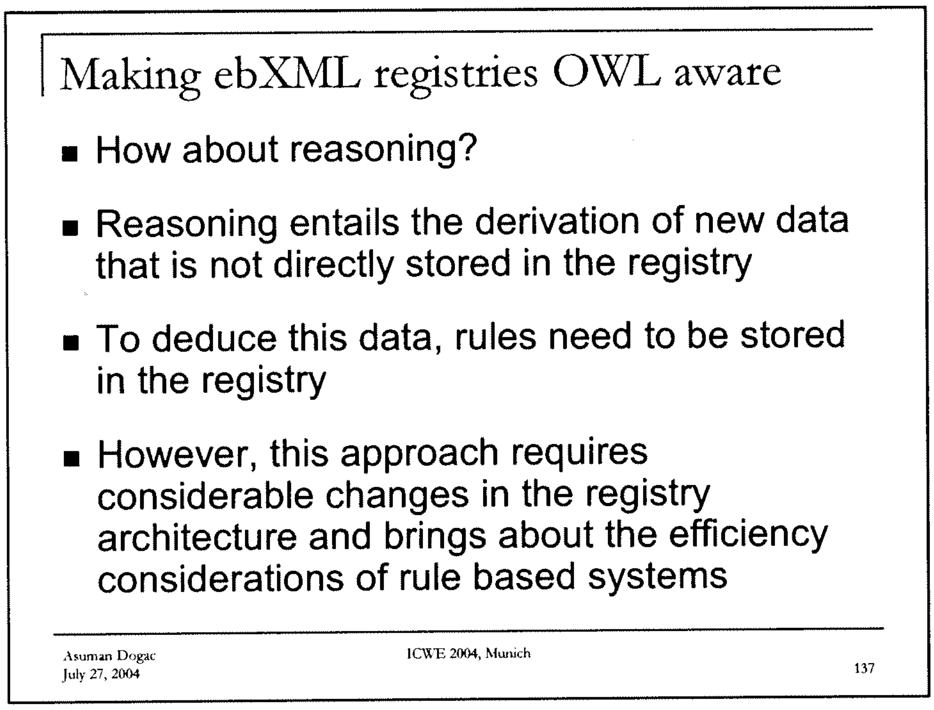
- For example, to find the super classes of a given class (defined through a new association type of “subClassOf”), a stored procedure can be defined
- The user can call this procedure when the need arises
- Furthermore, the stored procedures can also be called transparently to the user by changing only the query manager component of the registry

Another Example: Finding the ObjectProperties of a Class

```
CREATE PROCEDURE
  findObjectProperties($className) AS
BEGIN
SELECT A.id
FROM Association A, Name_ N, ClassificationNode C
WHERE A.associationType LIKE 'objectProperty' AND
    C.id = N.parent AND
    N.value LIKE $className AND
    A.sourceObject = C.id
END;
```

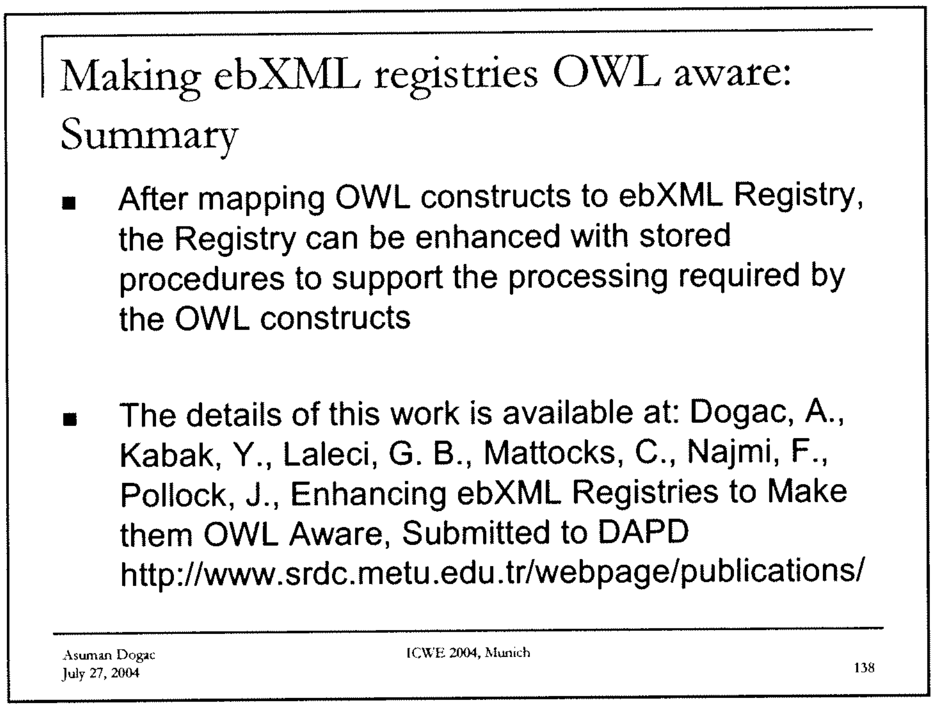
Another Example


OTA_AirReservation Services
IMHO_AirReservation Services
equivalentTo
Web Service 1
Web Service 2
Results returned by ebXML query
Result Returned by the given Stored Procedure
Web Service 3



Making ebXML registries OWL aware

- How about reasoning?
- Reasoning entails the derivation of new data that is not directly stored in the registry
- To deduce this data, rules need to be stored in the registry
- However, this approach requires considerable changes in the registry architecture and brings about the efficiency considerations of rule based systems



Making ebXML registries OWL aware: Summary

- After mapping OWL constructs to ebXML Registry, the Registry can be enhanced with stored procedures to support the processing required by the OWL constructs
- The details of this work is available at: Dogac, A., Kabak, Y., Laleci, G. B., Mattocks, C., Najmi, F., Pollock, J., Enhancing ebXML Registries to Make them OWL Aware, Submitted to DAPD http://www.srdc.metu.edu.tr/webpage/publications/

Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- ➡ Exploiting Web Service semantics in healthcare domain
  - Challenges in the Healthcare Informatics
  - IHE: Integrating Healthcare Enterprise
  - Artemis Project

Outline

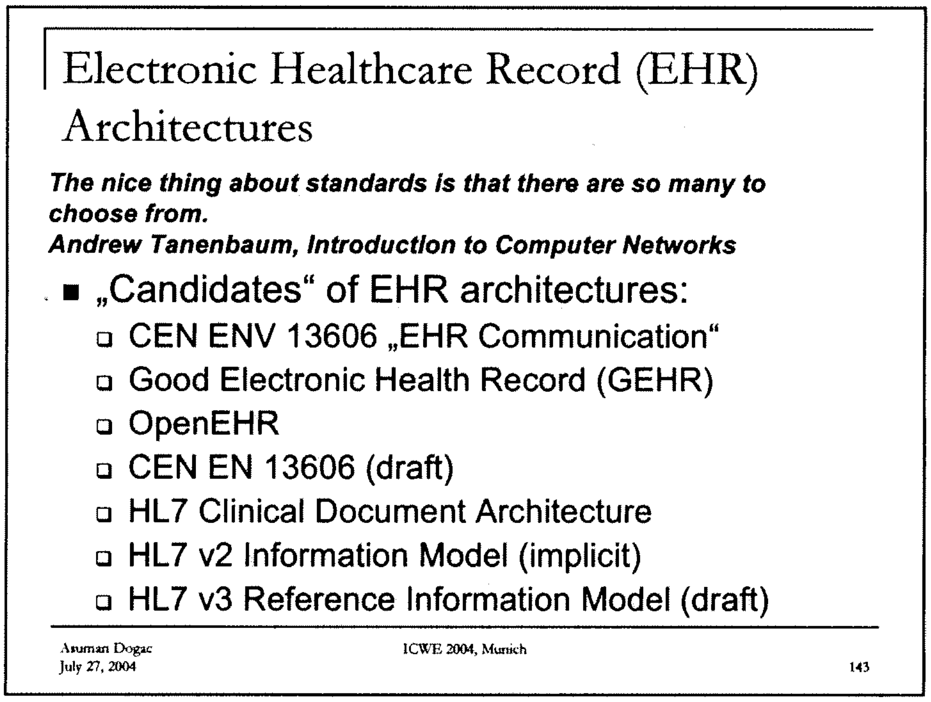
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
  - ➡ Challenges in the Healthcare Informatics
  - IHE: Integrating Healthcare Enterprise
  - Artemis Project

Challenges of Healthcare Informatics

- Most of the health information systems today are proprietary
- They often only serve one specific department within a healthcare institute
- To complicate the matters worse, a patient's health information may be spread out over a number of different institutes which do not interoperate
- This makes it very difficult for clinicians to capture a complete clinical history of a patient

Challenges of Healthcare Informatics

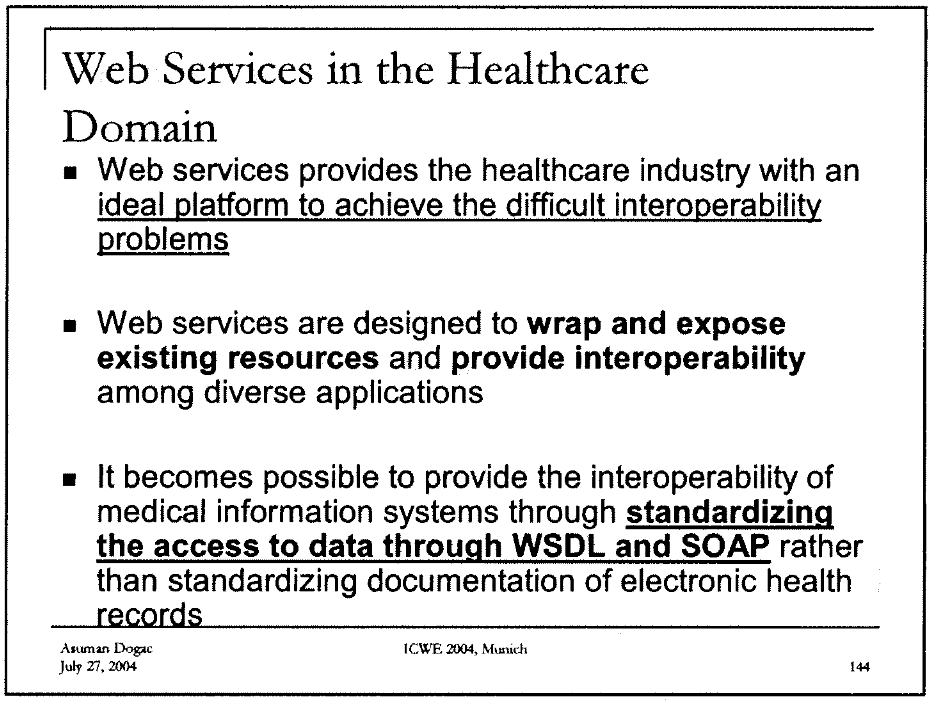
- The systems must interoperate for effectiveness
- For interoperability standards are needed
- However there are more than one standard in the health care domain



Electronic Healthcare Record (EHR) Architectures

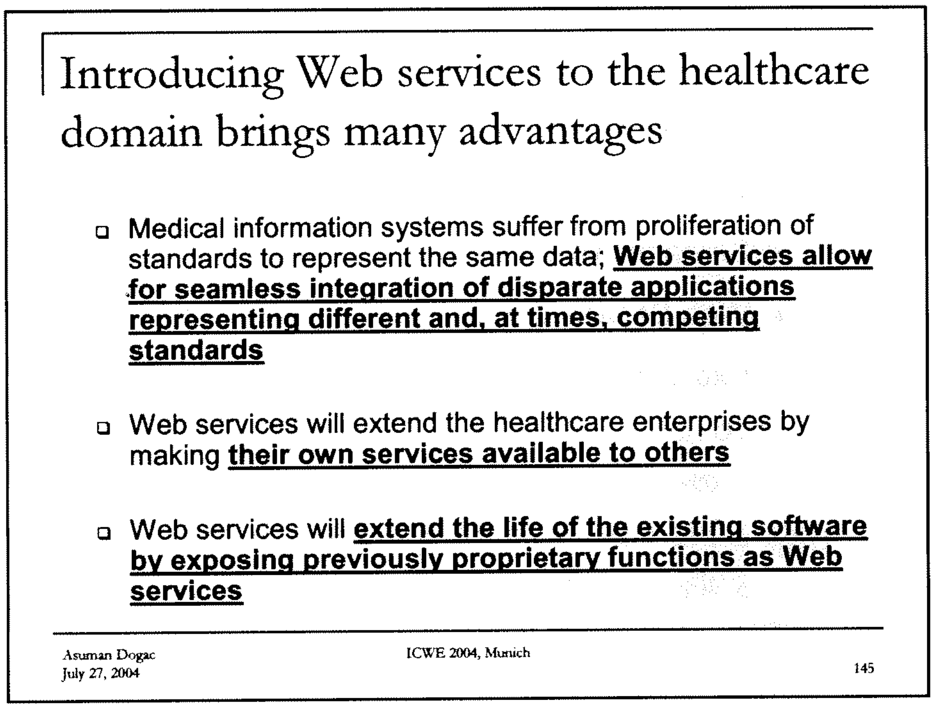
The nice thing about standards is that there are so many to choose from.
Andrew Tanenbaum, Introduction to Computer Networks

- „Candidates“ of EHR architectures:
  - CEN ENV 13606 „EHR Communication“
  - Good Electronic Health Record (GEHR)
  - OpenEHR
  - CEN EN 13606 (draft)
  - HL7 Clinical Document Architecture
  - HL7 v2 Information Model (implicit)
  - HL7 v3 Reference Information Model (draft)



Web Services in the Healthcare Domain

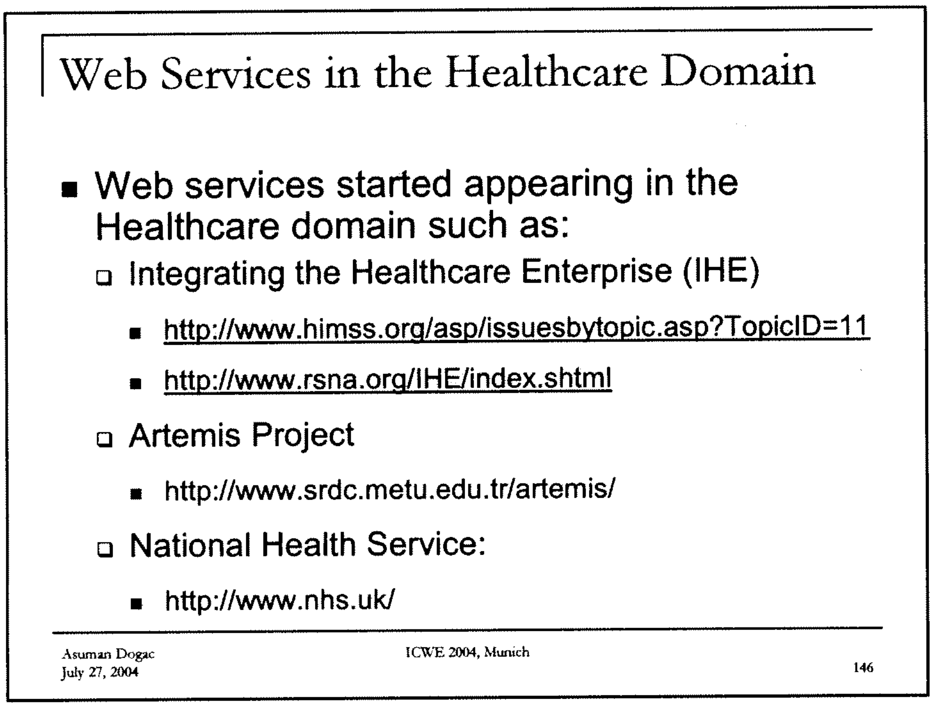
- Web services provides the healthcare industry with an <u>ideal platform to achieve the difficult interoperability problems</u>
- Web services are designed to **wrap and expose existing resources** and **provide interoperability** among diverse applications
- It becomes possible to provide the interoperability of medical information systems through **<u>standardizing the access to data through WSDL and SOAP</u>** rather than standardizing documentation of electronic health records



Introducing Web services to the healthcare domain brings many advantages

- Medical information systems suffer from proliferation of standards to represent the same data; **Web services allow for seamless integration of disparate applications representing different and, at times, competing standards**
- Web services will extend the healthcare enterprises by making **their own services available to others**
- Web services will **extend the life of the existing software by exposing previously proprietary functions as Web services**

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
145



Web Services in the Healthcare Domain

- Web services started appearing in the Healthcare domain such as:
  - Integrating the Healthcare Enterprise (IHE)
    - http://www.himss.org/asp/issuesbytopic.asp?TopicID=11
    - http://www.rsna.org/IHE/index.shtml
  - Artemis Project
    - http://www.srdc.metu.edu.tr/artemis/
  - National Health Service:
    - http://www.nhs.uk/

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
146

Outline

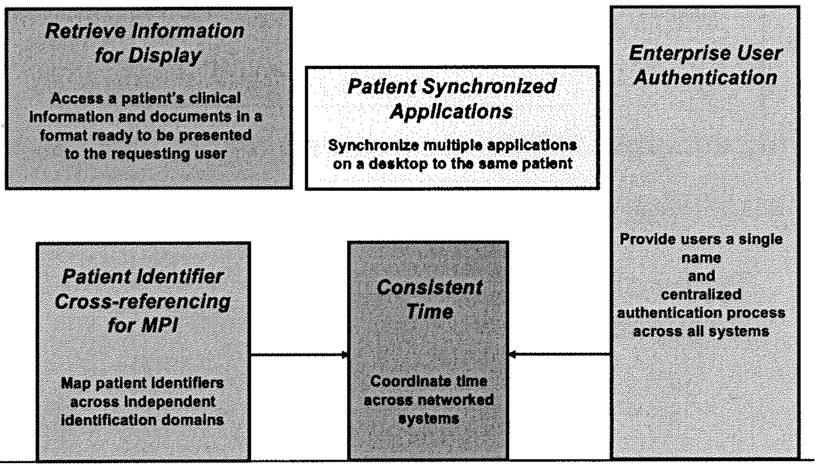
- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
  - Challenges in the Healthcare Informatics
  - ➡ IHE: Integrating Healthcare Enterprise
  - Artemis Project

iHE

Overview of IHE IT Infrastructure Integration Profiles

IHE IT Infrastructure Technical Committee
Charles Parisot, GE Medical Systems Information Technologies

IHE IT Infrastructure Defines 5 Integration Profiles


Retrieve Information for Display
Access a patient's clinical information and documents in a format ready to be presented to the requesting user
Patient Synchronized Applications
Synchronize multiple applications on a desktop to the same patient
Enterprise User Authentication
Provide users a single name and centralized authentication process across all systems
Patient Identifier Cross-referencing for MPI
Map patient identifiers across independent identification domains
Consistent Time
Coordinate time across networked systems

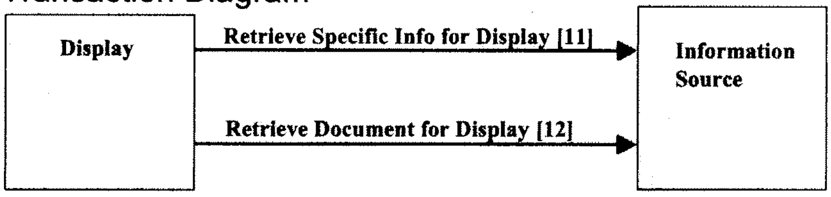
An Example: Retrieve Information for Display

Key Technical Properties:

- **Standards Used:**
  - **Web Services (WSDL for HTTP Get).**
  - General purpose IT Presentation Formats: XHTML, PDF, JPEG plus CDA L1.
  - Client may be off-the-shelf browser or display app.
- Two services :
  - Retrieve of Specific Information:
    - Patient centric: patient ID
    - Type of Request
    - Date, Time, nMostRecent
  - Retrieve a Document
    - Object Unique Instance Identifier (OID)
    - Type of Request
    - Content Type Expected

Retrieve Information for Display

Transaction Diagram


Display
Retrieve Specific Info for Display [11]
Retrieve Document for Display [12]
Information Source

Types of Requests

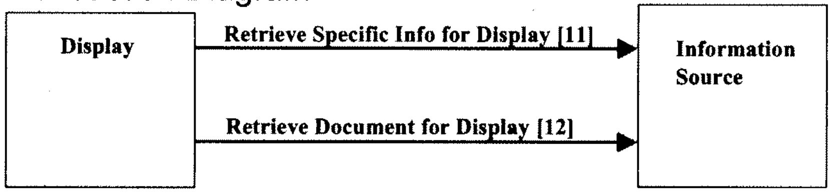
- Summary of All Reports
- Summary of Laboratory Reports
- Summary of Radiology Reports
- Summary of Cardiology Reports
- Summary of Surgery Reports
- Summary of Intensive Care Reports
- Summary of Emergency Reports
- Summary of Discharge Reports
- List of Allergies
- List of Medications
- Persistent Document

Summary: IHE

- IHE has defined a few basic Web services
- Yet, since IHE does not address semantic issues: to use IHE Web services, it is necessary to conform to their specification exactly,
  - by calling the Web services with the names they have specified, and
  - providing the messages as instructed in its specification

Retrieve Information for Display

Transaction Diagram


Display
Retrieve Specific Info for Display [11]
Retrieve Document for Display [12]
Information Source

Types of Requests

Summary of All Reports
Summary of Laboratory Reports
Summary of Radiology Reports
Summary of Cardiology Reports
Summary of Surgery Reports
Summary of Intensive Care Reports
Summary of Emergency Reports
Summary of Discharge Reports
List of Allergies
List of Medications
Persistent Document

Summary: IHE

- IHE has defined a few basic Web services
- Yet, since IHE does not address semantic issues: to use IHE Web services, it is necessary to conform to their specification exactly,
  - by calling the Web services with the names they have specified, and
  - providing the messages as instructed in its specification

Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
  - Challenges in the Healthcare Informatics
  - IHE: Integrating Healthcare Enterprise
  - Artemis Project

Artemis Project

artemis

A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information Systems
(IST-1-002103-STP)

Artemis Architecture

- The Artemis project addresses the interoperability problem in the healthcare domain where organisations have proprietary application systems to access data

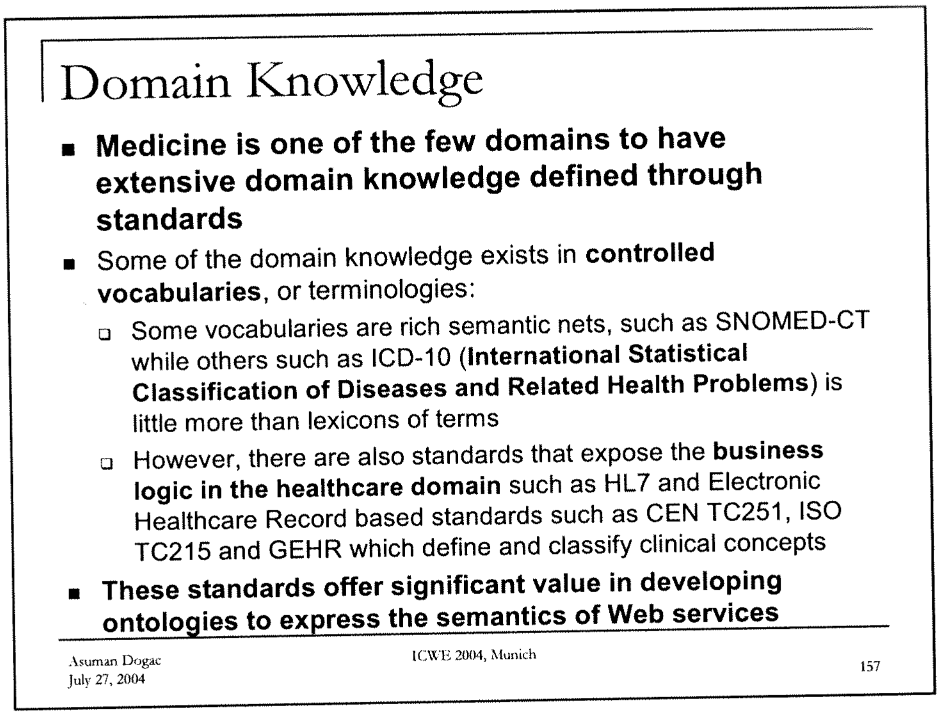
- To exchange information in an interoperable manner, the medical institutes:
  - Classify the Web services that they are providing through **Service Functionality Ontologies**
  - Determine the semantics of Service Messages through **Service Message Ontologies**

Semantic Mediation: Ontology Mapping

- The differences between disparate Service Functionality and Service Message Ontologies will be resolved through Ontology Mapping

- Although we propose to develop ontologies based on the prominent healthcare standards, **the ontologies we are proposing is just to facilitate ontology mediation**

- It realistic to expect healthcare institutes to conform to one global ontology



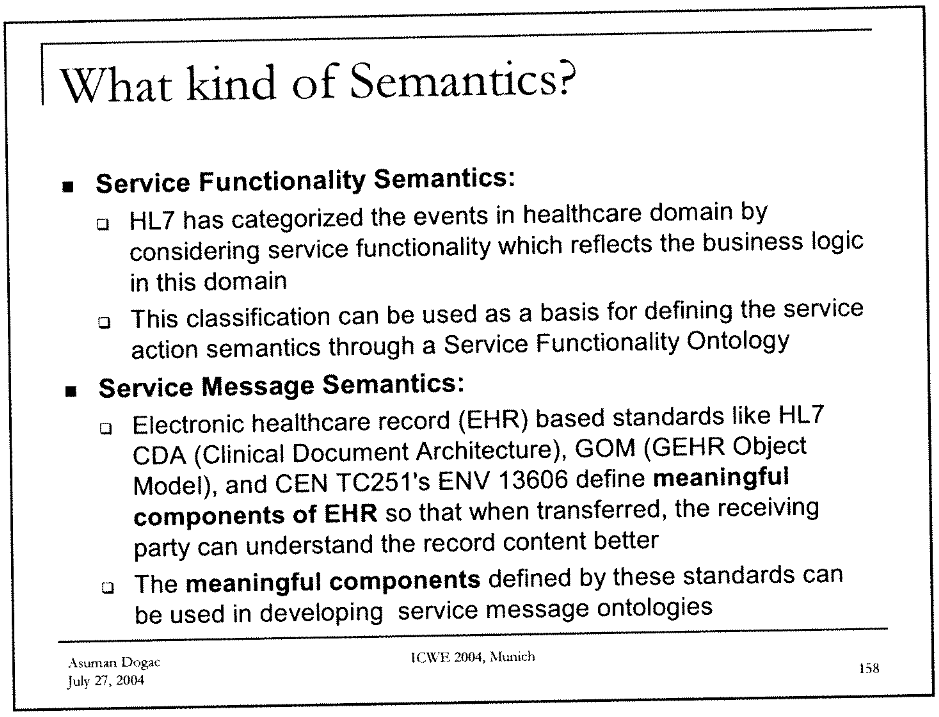
Domain Knowledge

- **Medicine is one of the few domains to have extensive domain knowledge defined through standards**
- Some of the domain knowledge exists in **controlled vocabularies**, or terminologies:
  - Some vocabularies are rich semantic nets, such as SNOMED-CT while others such as ICD-10 (**International Statistical Classification of Diseases and Related Health Problems**) is little more than lexicons of terms
  - However, there are also standards that expose the **business logic in the healthcare domain** such as HL7 and Electronic Healthcare Record based standards such as CEN TC251, ISO TC215 and GEHR which define and classify clinical concepts
- **These standards offer significant value in developing ontologies to express the semantics of Web services**

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

157



What kind of Semantics?

- **Service Functionality Semantics:**
  - HL7 has categorized the events in healthcare domain by considering service functionality which reflects the business logic in this domain
  - This classification can be used as a basis for defining the service action semantics through a Service Functionality Ontology
- **Service Message Semantics:**
  - Electronic healthcare record (EHR) based standards like HL7 CDA (Clinical Document Architecture), GOM (GEHR Object Model), and CEN TC251's ENV 13606 define **meaningful components of EHR** so that when transferred, the receiving party can understand the record content better
  - The **meaningful components** defined by these standards can be used in developing service message ontologies

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

158

HL7 and Web Services

- The primary goal of HL7 is to provide standards for the exchange of data among healthcare computer applications
- An event in the healthcare world, called the trigger event, causes exchange of messages between a pair of applications
- When an event occurs in an HL7 compliant system, an HL7 message is prepared by collecting the necessary data from the underlying systems and it is passed to the requestor, usually as an EDI message
- Mapping HL7's message based events directly into Web services may result in several inefficiencies

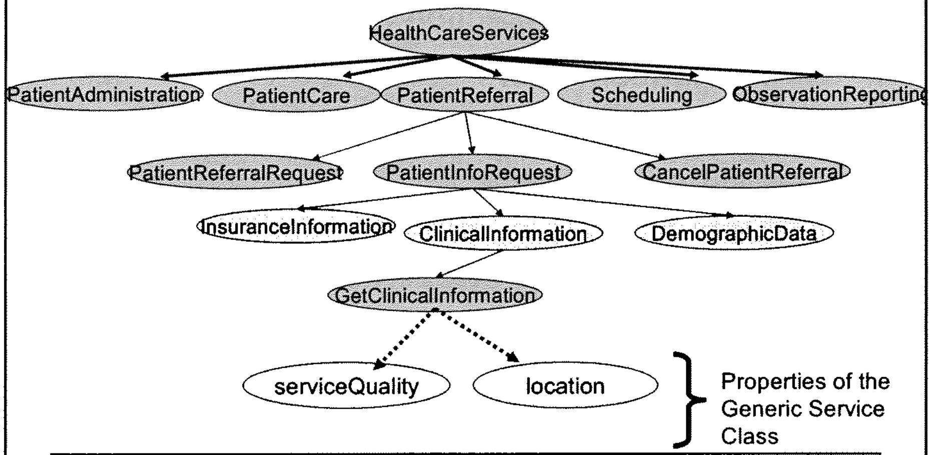
HL7 and Web Services

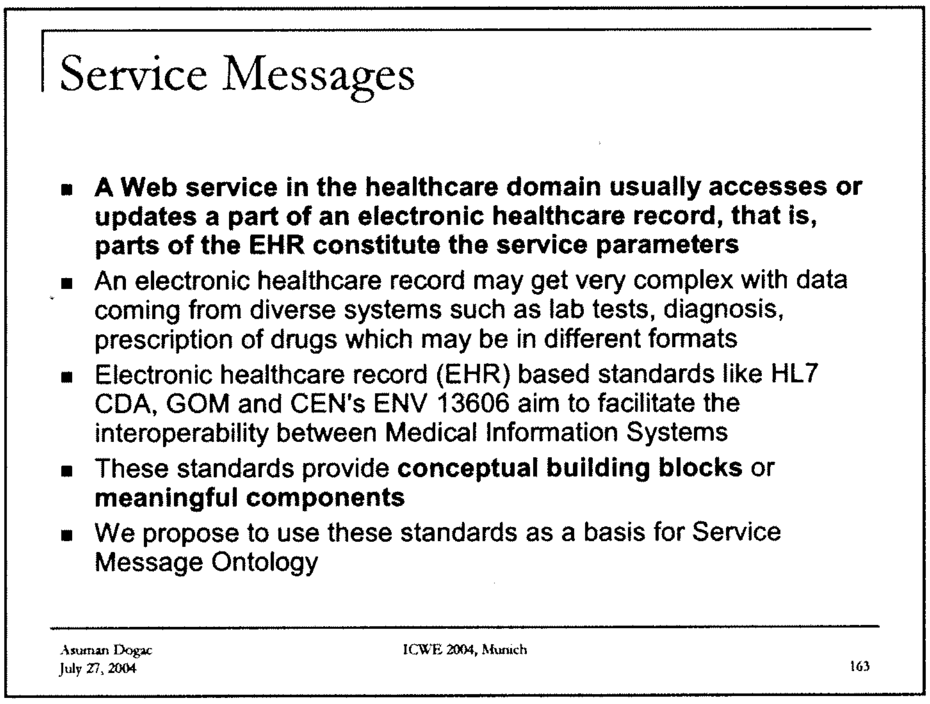
- The input and output messages defined for HL7 events are usually very **complex containing innumerous segments of different types and optionality**
- Furthermore, **all the semantics about the business logic and the document structure are hard coded in the message**
- This implies that, <u>the party invoking the Web service must be HL7 compliant to make any sense of the content of the output parameter(s) returned by the service</u>
- Furthermore, the information contained in an **HL7 message may be coming from different systems either proprietary or complying to different standards**
- Hence, in Web services terminology, HL7 events correspond to **Composite services**, whereas more elementary services are needed

HL7 and Web Services

- Since HL7 has already been through an effort of categorizing the events in healthcare domain considering service functionality, it can be used as a basis for a service functionality ontology

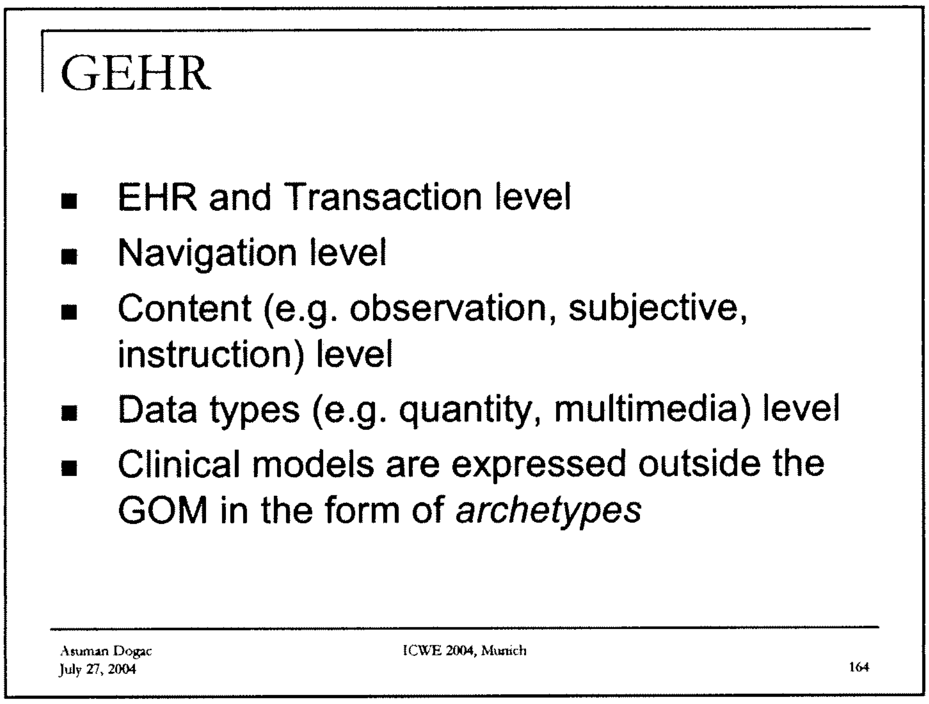
An Example Service Functionality Ontology





Service Messages

- **A Web service in the healthcare domain usually accesses or updates a part of an electronic healthcare record, that is, parts of the EHR constitute the service parameters**
- An electronic healthcare record may get very complex with data coming from diverse systems such as lab tests, diagnosis, prescription of drugs which may be in different formats
- Electronic healthcare record (EHR) based standards like HL7 CDA, GOM and CEN's ENV 13606 aim to facilitate the interoperability between Medical Information Systems
- These standards provide **conceptual building blocks** or **meaningful components**
- We propose to use these standards as a basis for Service Message Ontology



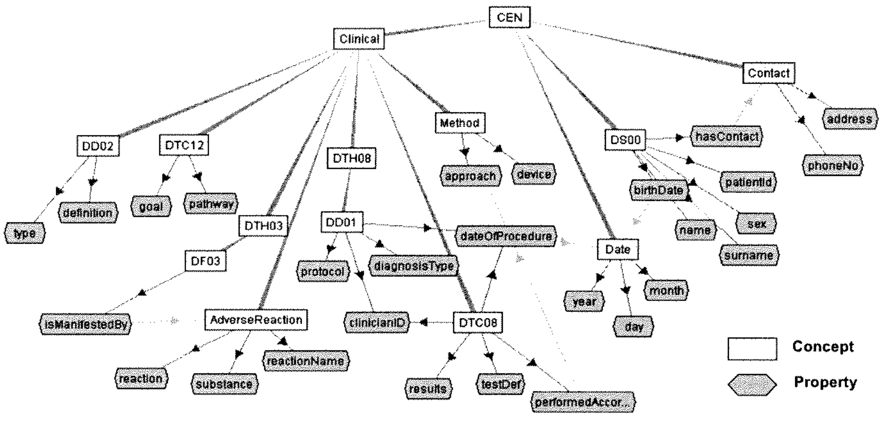
GEHR

- EHR and Transaction level
- Navigation level
- Content (e.g. observation, subjective, instruction) level
- Data types (e.g. quantity, multimedia) level
- Clinical models are expressed outside the GOM in the form of *archetypes*

CEN TC 251 ENV 13606

- *Folder:* High-level subdivisions of the entire EHR for a patient
- *Composition:* A set of record entries relating to one time and place of care delivery; grouped contributions to an aspect of health care activity; composed reports and overviews of clinical progress
- *Headed Section:* Sub-divisions used to group entries with a common theme or derived through a common healthcare process.
- *Cluster:* Low-level aggregations of elementary entries (Record Items) to represent a compound clinical concept

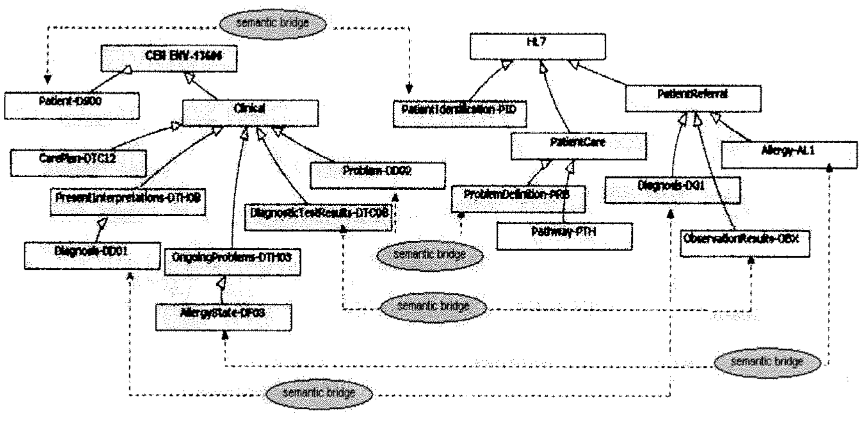
An example Service Message Ontology



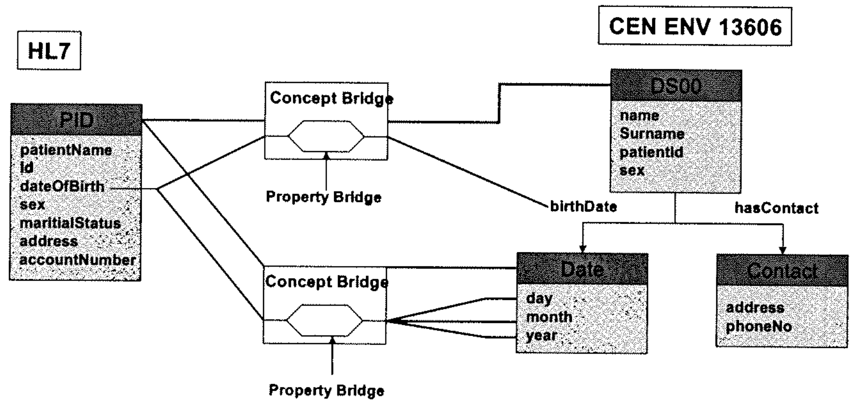
Semantic Mediation

- In Artemis architecture, the healthcare institutes can develop their own ontologies
- However these ontologies are based on standards developed by the healthcare standardization bodies like CEN TC251, ISO TC215, GEHR or HL7
- The ontology mappings are achieved through semantic mediation

Mapping Message Ontologies



Semantic Mediation through MAFRA Tool

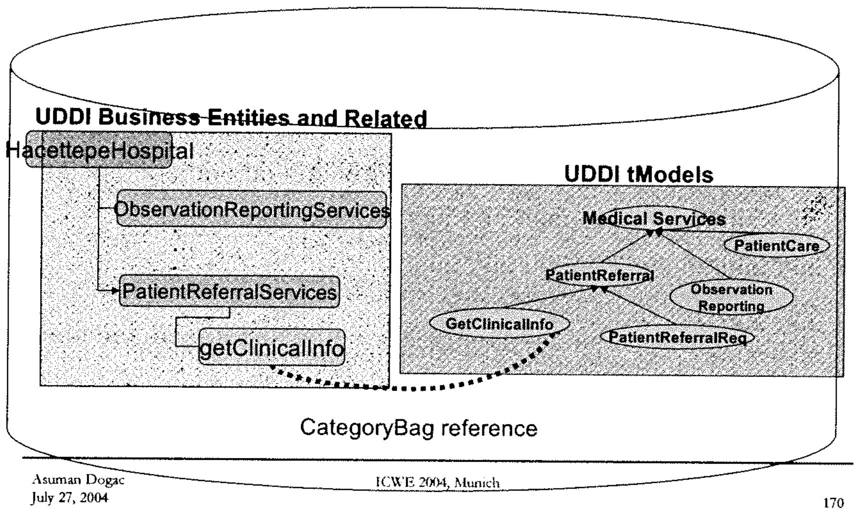


Asuman Dogac
July 27, 2004

ICWE 2004, Munich

169

Relating the services with the semantic defined through an ontology - UDDI

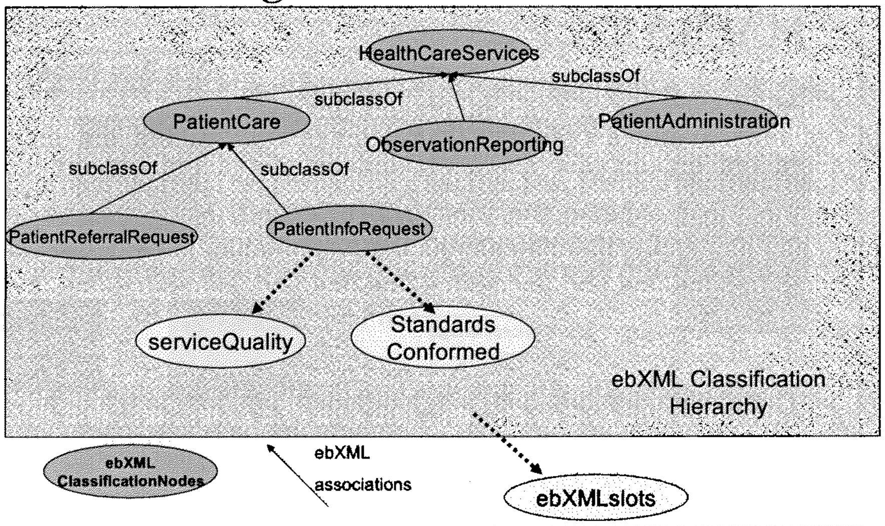


Asuman Dogac
July 27, 2004

ICWE 2004, Munich

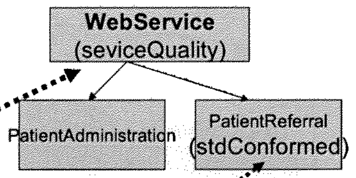
170

Associating semantics to ebXML

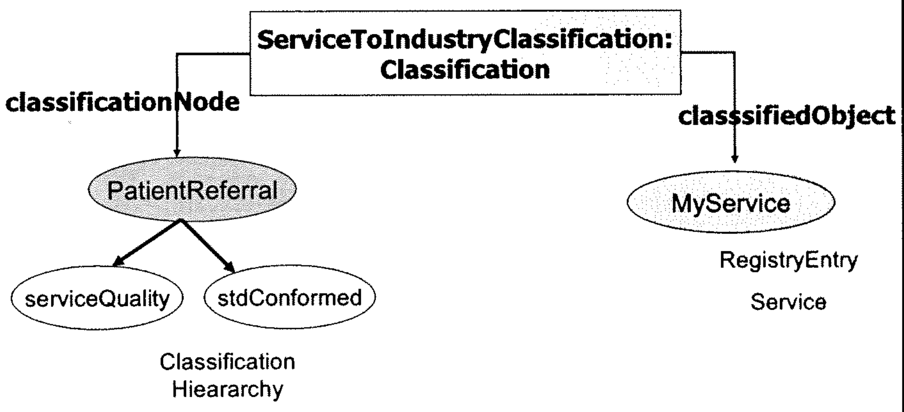


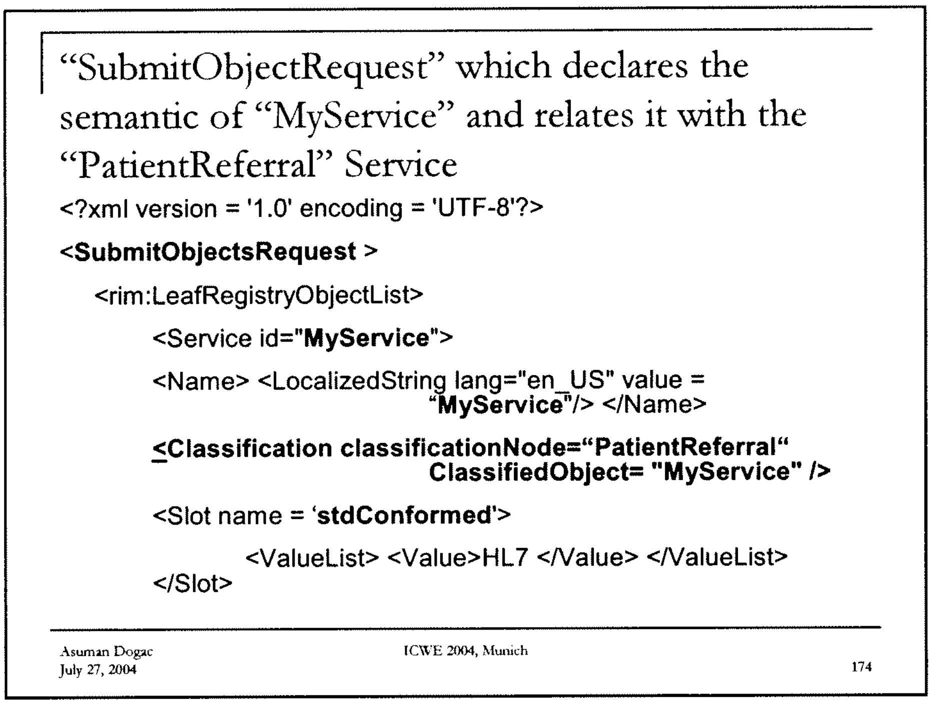
How to Define a Classification Hierarchy in ebXML?

```
<rim:ClassificationScheme id = 'WebService'
    isInternal='true' nodeType='UniqueCode' >
  <rim:Name>
      <rim:LocalizedString value = 'WebService'/>
    </rim:Name>
    <rim:Description>
<rim:LocalizedString value = 'This is a
          sample WebServicescheme'/>
    </rim:Description>
    <Slot name = 'serviceQuality' slotType=
                                     'StringList'/>
</rim:ClassificationScheme>
        <rim:ClassificationNode id = 'PatientReferral'  parent= 'WebService'>
                  <rim:Name>
                        <rim:LocalizedString value = 'PatientReferral' /> </rim:Name>
                  <rim:Description>
                         <rim:LocalizedString value = ''/>
                   </rim:Description>
                     <Slot name = 'stdConformed' slotType= 'StringList'/>
        </rim:ClassificationNode> …
```



Relating the services with the semantic defined through an ontology - ebXML





"SubmitObjectRequest" which declares the semantic of "MyService" and relates it with the "PatientReferral" Service

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SubmitObjectsRequest >
  <rim:LeafRegistryObjectList>
      <Service id="MyService">
      <Name> <LocalizedString lang="en_US" value =
                         "MyService"/> </Name>
      <Classification classificationNode="PatientReferral"
                         ClassifiedObject= "MyService" />
      <Slot name = 'stdConformed'>
              <ValueList> <Value>HL7 </Value> </ValueList>
      </Slot>
```

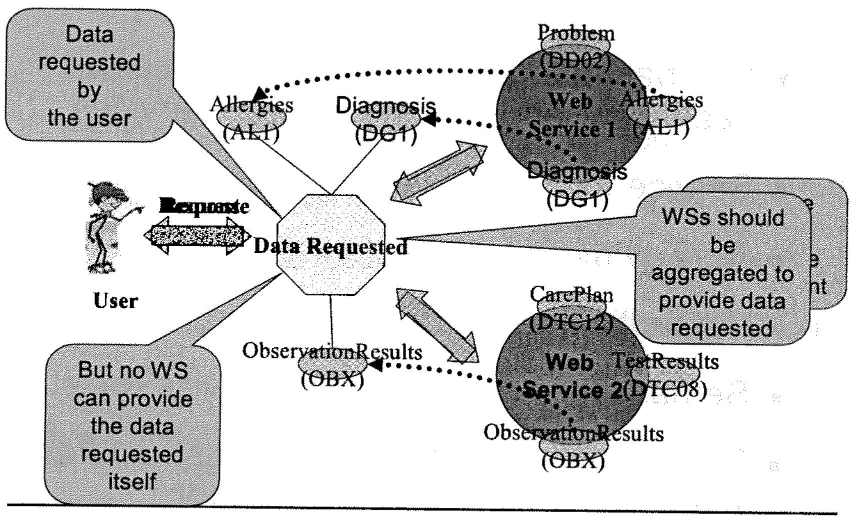
An Example "SubmitObjectRequest" (Cont'd)

```
        <ServiceBinding
        accessURI="http://www.sun.com/ebxmlrr/registry/nameSpaceI
                      ndexer">
        <SpecificationLink specificationObject="wsdl">
        </SpecificationLink> </ServiceBinding> </Service>

        <ExtrinsicObject id="wsdl" mimeType="text/xml">
        </ExtrinsicObject>
    </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

Web Service Aggregation

- Although classifying the Web Services through the semantic category of the data they are providing facilitates the discovery of the services fetching a specific part of EHR data, **it may not always be possible to find a service delivering exactly the data requested**
- For example, a healthcare institute may be providing the diagnosis information as a part of another clinical concept
- This may necessitate more complex aggregations of Web Services
- We address how complex aggregation of Web services can be handled by taking advantage of the ontology mappings

An Example to Service Aggregation

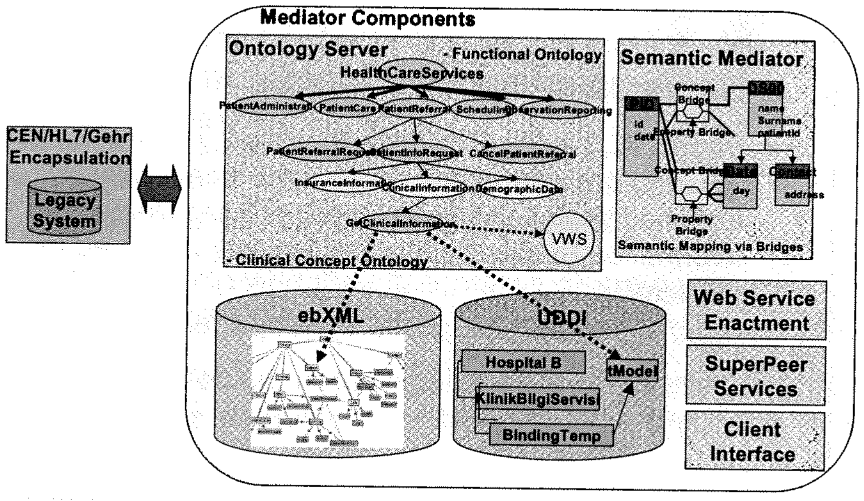


Asuman Dogac
July 27, 2004

ICWE 2004, Munich

177

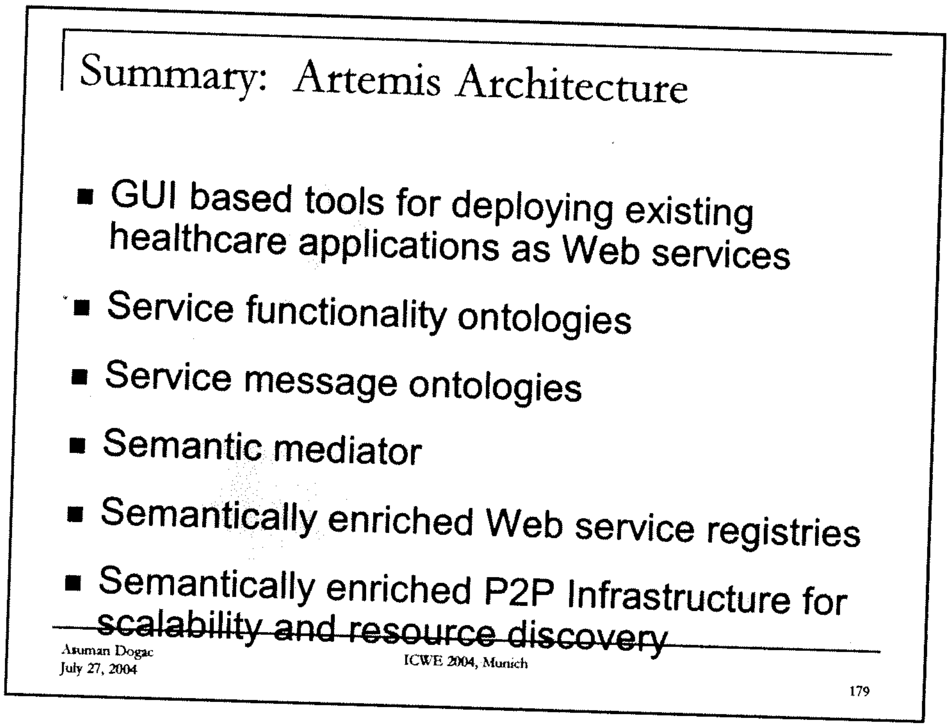
An Overview of a Mediator



Asuman Dogac
July 27, 2004

ICWE 2004, Munich

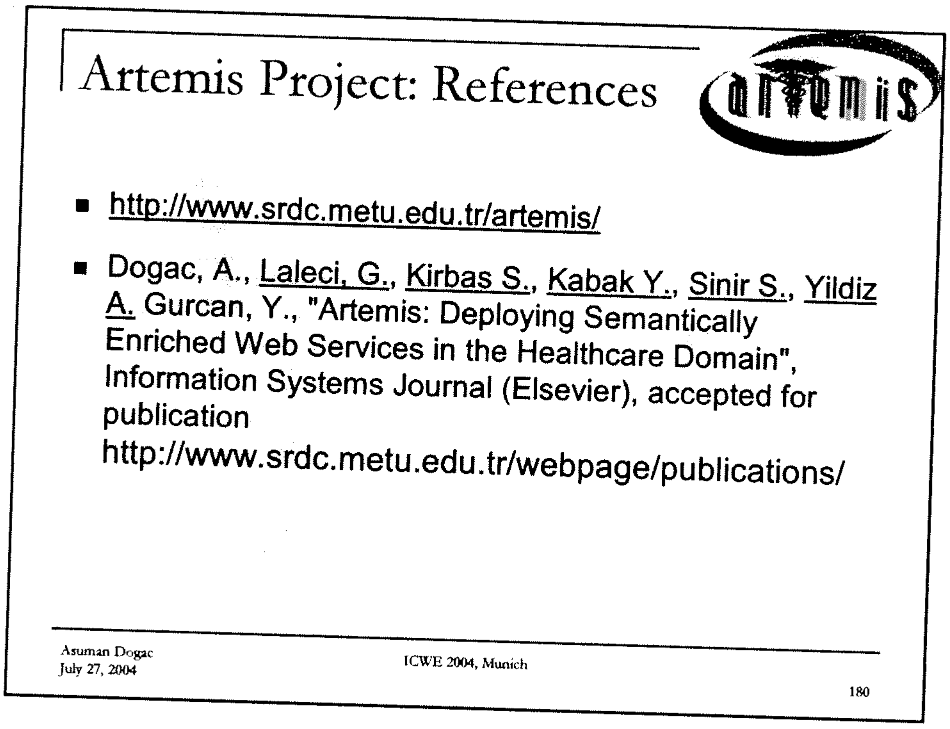
178



Summary: Artemis Architecture

- GUI based tools for deploying existing healthcare applications as Web services
- Service functionality ontologies
- Service message ontologies
- Semantic mediator
- Semantically enriched Web service registries
- Semantically enriched P2P Infrastructure for scalability and resource discovery

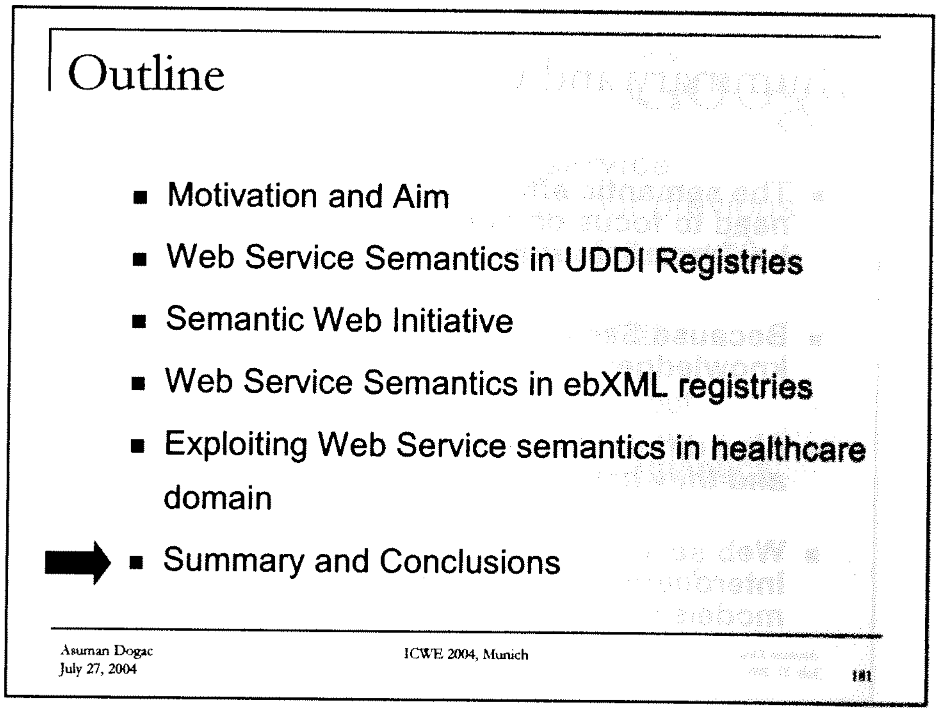
Asuman Dogac
July 27, 2004
ICWE 2004, Munich
179



Artemis Project: References

- http://www.srdc.metu.edu.tr/artemis/
- Dogac, A., Laleci, G., Kirbas S., Kabak Y., Sinir S., Yildiz A. Gurcan, Y., "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain", Information Systems Journal (Elsevier), accepted for publication http://www.srdc.metu.edu.tr/webpage/publications/

Asuman Dogac
July 27, 2004
ICWE 2004, Munich
180



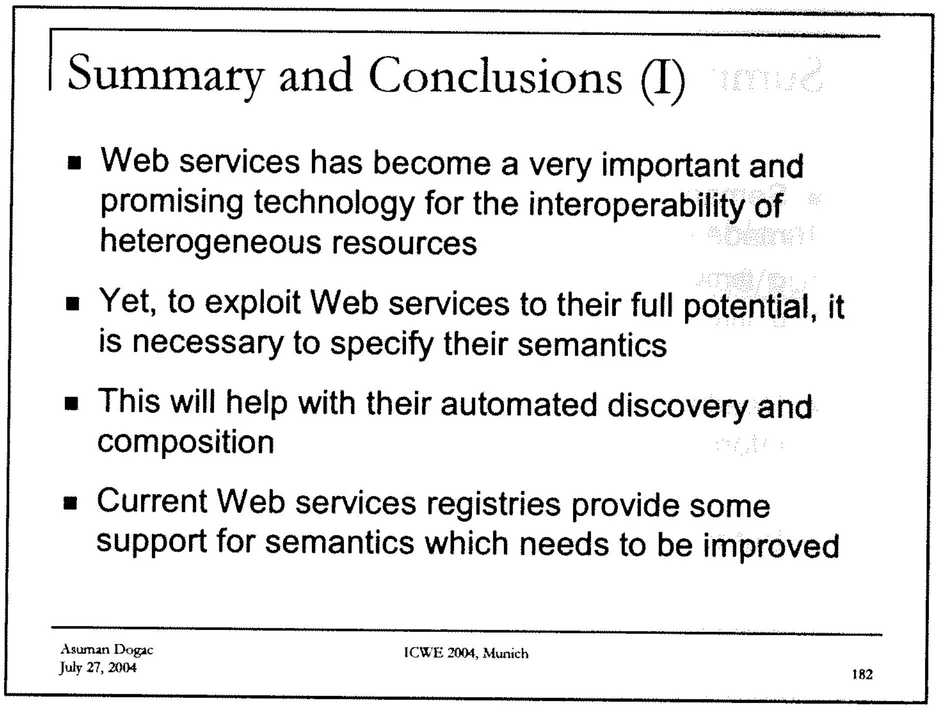
Outline

- Motivation and Aim
- Web Service Semantics in UDDI Registries
- Semantic Web Initiative
- Web Service Semantics in ebXML registries
- Exploiting Web Service semantics in healthcare domain
- ➡ Summary and Conclusions

Asuman Dogac
July 27, 2004

ICWE 2004, Munich

181



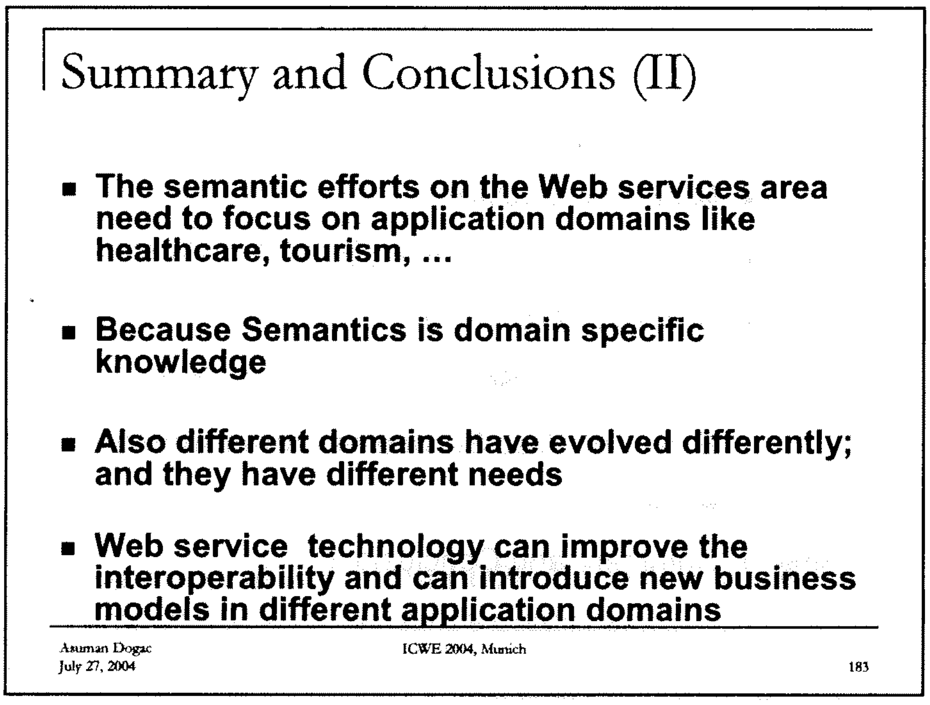
Summary and Conclusions (I)

- Web services has become a very important and promising technology for the interoperability of heterogeneous resources
- Yet, to exploit Web services to their full potential, it is necessary to specify their semantics
- This will help with their automated discovery and composition
- Current Web services registries provide some support for semantics which needs to be improved

Asuman Dogac
July 27, 2004

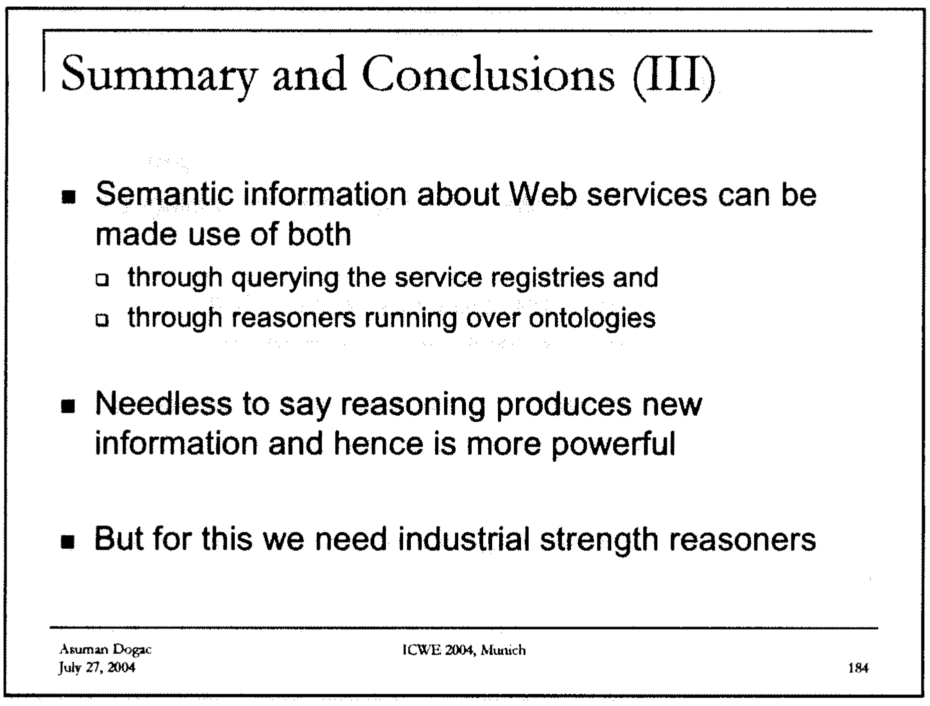
ICWE 2004, Munich

182



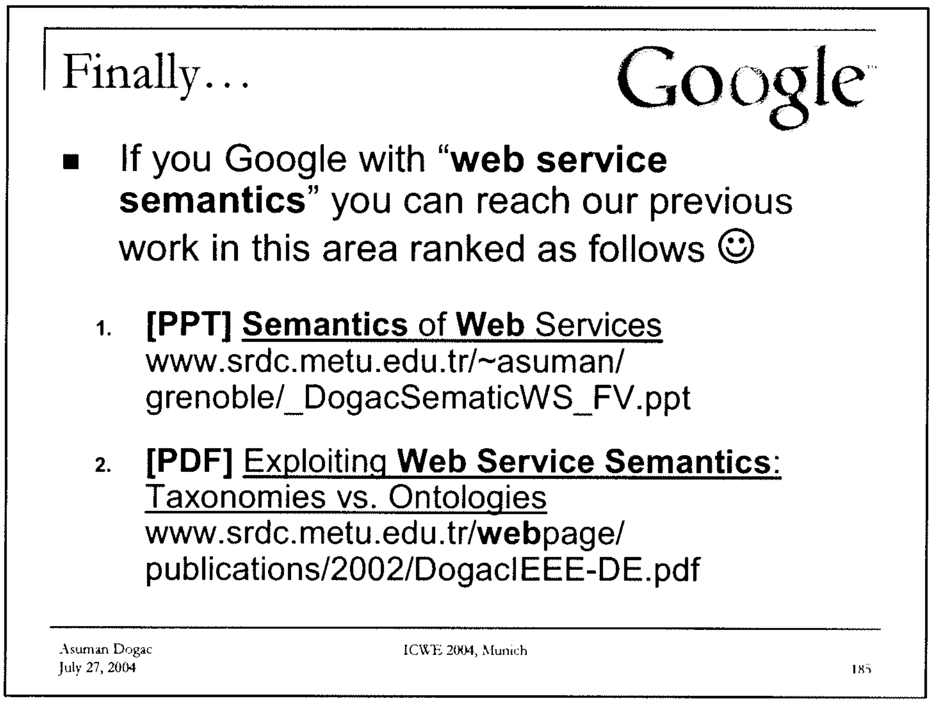
Summary and Conclusions (II)

- **The semantic efforts on the Web services area need to focus on application domains like healthcare, tourism, ...**
- **Because Semantics is domain specific knowledge**
- **Also different domains have evolved differently; and they have different needs**
- **Web service technology can improve the interoperability and can introduce new business models in different application domains**



Summary and Conclusions (III)

- Semantic information about Web services can be made use of both
  - through querying the service registries and
  - through reasoners running over ontologies
- Needless to say reasoning produces new information and hence is more powerful
- But for this we need industrial strength reasoners

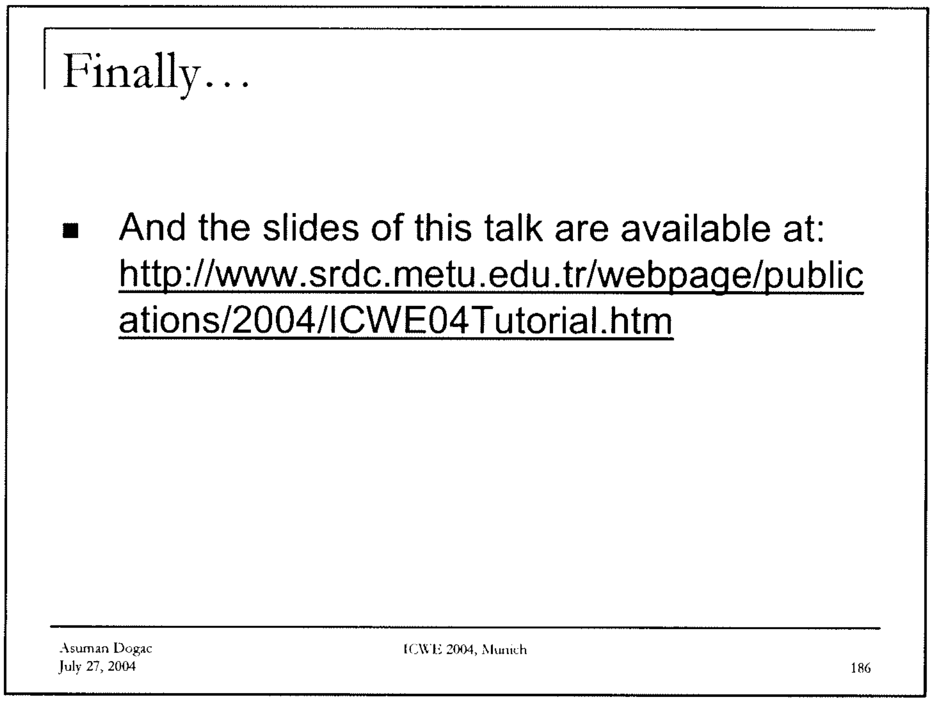


Finally… Google™

- If you Google with "**web service semantics**" you can reach our previous work in this area ranked as follows ☺

1. **[PPT]** **Semantics** of **Web** Services
   www.srdc.metu.edu.tr/~asuman/grenoble/_DogacSematicWS_FV.ppt
2. **[PDF]** Exploiting **Web Service Semantics**: Taxonomies vs. Ontologies
   www.srdc.metu.edu.tr/**web**page/publications/2002/DogacIEEE-DE.pdf

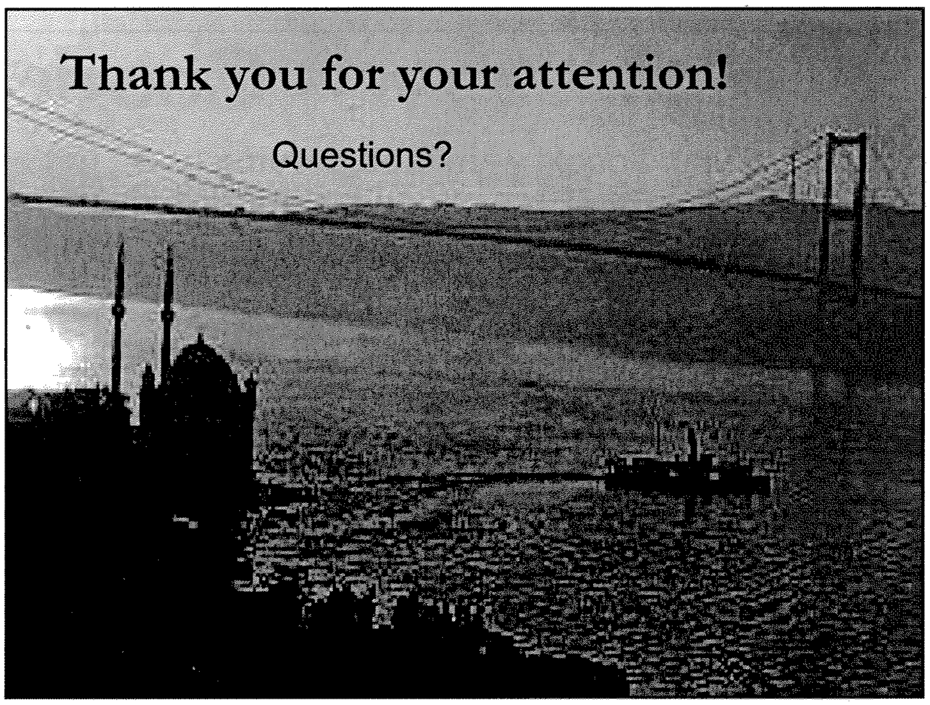
Asuman Dogac
July 27, 2004 ICWE 2004, Munich 185



Finally…

- And the slides of this talk are available at: http://www.srdc.metu.edu.tr/webpage/publications/2004/ICWE04Tutorial.htm

Asuman Dogac
July 27, 2004 ICWE 2004, Munich 186



Thank you for your attention!

Questions?