EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

ANLAMSAL WEB İÇİN KİŞİSELLEŞTİRİLEBİLİR ONTOLOJİ TABANLI ERİŞİM DENETİMİ VE POLİTİKA YÖNETİMİ

Özgü CAN

Bilgisayar Mühendisliği Anabilim Dalı
Bilim Dalı Kodu: 619.01.00
Sunuş Tarihi: 03.09.2009

Tez Danışmanı: Yard. Doç. Dr. Murat Osman ÜNALIR

Bornova - İZMİR

II

Özgü CAN tarafından **DOKTORA** tezi olarak sunulan "Anlamsal Web İçin Kişiselleştirilebilir Ontoloji Tabanlı Erişim Denetimi ve Politika Yönetimi" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 03.09.2009 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri: **İmza**

Jüri Başkanı : Yrd. Doç. Dr. Murat Osman ÜNALIR ……...……
Raportör Üye : Yrd. Doç. Dr. Murat KOMESLİ ……...……
Üye : Prof. Dr. Alp KUT ……...……
Üye : Prof. Dr. Halil ŞENGONCA ……...……
Üye : Prof. Dr. Oğuz DİKENELLİ ……...……

ÖZET

ANLAMSAL WEB İÇİN KİŞİSELLEŞTİRİLEBİLİR ONTOLOJİ TABANLI ERİŞİM DENETİMİ VE POLİTİKA YÖNETİMİ

CAN, Özgü
Doktora Tezi, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Murat Osman ÜNALIR
Eylül 2009, 260 sayfa

Anlamsal Web, bilginin paylaşılmasını ve yeniden kullanımını sağlamak için, biçimsel anlambilimini kullanarak makinelerin diğer makineler ile haberleşmesine izin vermektedir. Ancak, bilginin paylaşılması gizlilik, erişim denetimi, kimlik denetimi, yetki ve veri bütünlüğü gibi güvenlik gereksinimlerini getirmekte; böylece, Anlamsal Web teknolojilerinin güvenliğini sağlayacak etkili düzeneklere gereksinim duyulmaktadır. Politika, bir kaynak için o kaynakta herhangi bir değişiklik yapılmadan bir erişim denetim düzeneğinin gerçekleştirilmesini sağlamaktadır. Bu tezde bilgiye erişimin ontoloji tabanlı bir yaklaşım ile gerçekleştirildiği bir Ontoloji Tabanlı Erişim Denetimi (Ontology Based Access Control - OBAC) önerilmektedir. Bu yaklaşımda, ontolojiler iki konuyu modellemek için kullanılmaktadır: (i) erişim denetim düzeneği (ii) kaynağın ve o kaynağa erişmek isteyen varlığın bilgisi. Geleneksel erişim denetim düzeneklerinde politikalar kaynağa erişimin denetiminde kullanılırken, bu tezde politikalar erişim denetiminde kişiselleştirmeyi gerçekleştirmek için seçeneklerin ve profillerin yönetiminde kullanılmaktadır. Bu tezde, kişiselleştirme için üç çeşit politika tanımlanmaktadır: etki alanı, profil ve seçenek politikaları.

Anahtar sözcükler: Anlamsal Web, Politika, Ontoloji, Kişiselleştirme.

ABSTRACT

PERSONALIZABLE ONTOLOGY BASED ACCESS CONTROL FOR SEMANTIC WEB AND POLICY MANAGEMENT

CAN, Özgü
Ph.D, Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Murat Osman ÜNALIR
September 2009, 260 pages

Semantic Web allows machines to share and reuse the information by using formal semantics to communicate with other machines. However, sharing the information brings security needs like privacy, access control, authentication, authorization and data integrity; thus, efficient mechanisms are needed to provide the security of Semantic Web technologies. A policy provides an access control mechanism for a resource without making any change in that resource. We propose an Ontology Based Access Control (OBAC) in which accessing the data is achieved by an ontology based approach. In this ontology based approach we use ontologies to model two issues: (i) model of the access control mechanism (ii) data of resource and entity which wants to access to that resource. In this work policies are used for the management of preferences and profiles in order to achieve access control for personalization, while in traditional access control mechanisms policies are used to control the access to a resource. Therefore, we defined three kinds of policies in this thesis: domain, profile and preference policies for a better personalization.

Keywords: Semantic Web, Policy, Ontology, Personalization.

TEŞEKKÜR

Bu tez çalışması süresince, deneyimleri ve önerilerinden yararlandığım danışmanım Sayın Yrd. Doç. Dr. Murat Osman ÜNALIR'a teşekkür ederim.

Gösterdikleri yakın ilgi ve yardımları için Ege Üniversitesi Bilgisayar Mühendisliği Bölüm Başkanı Sayın Prof. Dr. Halil ŞENGONCA ve Dokuz Eylül Üniversitesi Bilgisayar Mühendisliği Bölün Başkanı Sayın Prof. Dr. Alp KUT'a teşekkür ederim.

Her zaman yanımda olan ve beni destekleyen annem Mine CAN'a ve babam Fedai CAN'a teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR

Kısaltmalar	Açıklama
ABAC	Attribute Based Access Control
CWM	Closed World Machine
DAC	Discretionary Access Control
FIPA	Foundations of Intelligent Physical Agents
FOAF	Friend Of A Friend
HTML	Hyper Text Markup Language
JAS	Java Agent Services
JTP	Java Theorem Prover
KPAT	KAoS Policy Administration Tool
KPO	KAoS Policy Ontology
MAC	Mandatory Access Control
N3	Notation 3
OAP	Ontology Administration Point
OBAC	Ontology Based Access Control
OrBAC	Organization Based Access Control
OWL	Web Ontology Language
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Poliy Enforcement Point
P3P	Platform for Privacy Preferences
PIP	Policy Information Point

KISALTMALAR (devam)

Kısaltmalar	Açıklama
RBAC	Role Based Access Control
RDF	Resource Description Framework
RDFS	RDF Schema
RuleML	Rule Markup Language
SWRL	Semantic Web Rule Language
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WSPL	Web Services Policy Language
WWW	World Wide Web
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language

1. GİRİŞ

Bugünkü örgünün içeriği makinelerden çok insanlar için biçimlendirilmiştir. World Wide Web'in (WWW) yaratıcısı olan Tim Berners-Lee bugünkü örgü teknolojilerinin yetersizliklerini farkederek örgüyü daha zeki bir duruma getirmeye çalışmıştır. Bu amaçla makineler tarafından da anlaşılabilecek örgü sayfalarına ve bilginin bütünleştirilmesi için ontolojilerin kullanımına gereksinim olduğu sonucuna ulaşmıştır. Böylelikle, Anlamsal Web kavramı ortaya çıkmıştır.

Anlamsal Web, bilginin paylaşılmasını ve yeniden kullanımını sağlamak için, biçimsel anlambilimini kullanarak makinelerin diğer makineler ile iletişimine izin vermektedir. Böylece, bugünkü örgüde kullanıcılar örgü sayfalarını okuyup kararlarını vermekteyken, Anlamsal Web'de ise ortak ontolojiler ve betimleme dilleri kullanılarak kullanıcıları temsil eden etmenler örgü sayfalarını okuyup anlayabilir ve karar verebilirler. Etmen, kendisinden beklenenleri yerine getirmek için belli bir ortamda belli derecede özerklik çerçevesinde çalışan, algılayıcıları ile ortamdaki devingen değişimleri algılayan ve elde ettiği algılara göre bilgisini, amaçlarını yeniden değerlendiren, amaçları doğrultusunda planlama yaparak bu planlara ilişkin eylemleri yapan, diğer etmenler ile bir etmenler arası iletişim dili aracılığı ile iletişimde bulunma yeteneği olan ve bulunduğu ortamda süreklilik gösteren yazılım veya donanım tabanlı sistemdir (Erdur, 2001).

Ontolojiler varlıklar için ortak tanımlamalardır. Farklı terimleri

açıklamada ontolojilere gereksinim duyulduğundan örgü sayfalarının makineler tarafından anlaşılabilir olması için ontolojiler önemlidir.

Bilginin paylaşılması gizlilik, erişim denetimi, kimlik denetimi, yetki ve veri bütünlüğü gibi güvenlik gereksinimlerini getirmekte ve Anlamsal Web teknolojilerinin güvenliğini sağlayacak etkili düzeneklere gereksinim duyulmaktadır. Bu nedenle Anlamsal Web ile ilgili en temel çalışmalardan biri güvenilir Anlamsal Web'in (*Secure Semantic Web*) geliştirilmesidir. Güvenilir Anlamsal Web ile güvenlik gereksinimlerinin sağlanması amaçlanmaktadır. Çizelge 1.1 Anlamsal Web için Tim Berners-Lee tarafından geliştirilen katmanlı yapıyı göstermektedir, Çizelge 1.2'de ise güvenilir Anlamsal Web için her bir katmanda güvenliğin gerekli olduğu görülmektedir (Thuraisingham, 2007).

Çizelge 1.1 Anlamsal Web katmanlı yapısı.

Mantık, Kanıt ve Güven
Kurallar/Sorgu
RDF, Ontolojiler
XML, XML Şemalar
URI, UNICODE

Çizelge 1.2 Erişim denetimleri.

Güvenlik ile uyumlu Mantık, Kanıt ve Güven
Kurallar/Sorgu için Güvenlik
RDF, Ontolojiler için Güvenlik
XML, XML Şemalar için Güvenlik
URI, UNICODE için Güvenlik

Veriye ve bilgiye erişimin denetlenmesi gerekmektedir. Güvenilir dizgelerde verilere erişimin denetlenmesi ve bilginin yönetilmesi sağlanmaktadır. Bu amaçla politikalar kullanılmaktadır. Politika, bir kaynak için o kaynakta herhangi bir değişiklik yapılmadan bir erişim denetim düzeneğinin gerçekleştirilmesini sağlamaktadır. Erişim denetim düzeneklerinde "*Kullanıcı A, B1.doc dosyasını okuyabilir ve C1.doc dosyasına yazabilir.*" gibi olumlu kurallar ve "*Kullanıcı A, A1.doc dosyasına yazamaz ve D1.doc dosyasını okuyamaz.*" gibi olumsuz kurallar tanımlanabilmektedir. Bu kurallarda okumak ve yazmak kuralın eylemlerini oluştururken eylemin yapılıp yapılamaması ve hangi koşullar altında yapılacağının belirtilmesi kuralı oluşturmaktadır.

Anlamsal Web'de politika yönetimi bir kaynağa erişim için kuralların tanımlanması, kullanıcıların bu kuralları yorumlaması ve kurallara uyması için kullanılmaktadır. Anlamsal olarak zengin bir biçimde tanımlanmış olan politikalar, insan yanılgılarını ve politika çelişkilerini azaltmakta, politika çözümünü ve birlikte işlerliği kolaylaştırmaktadır (Tonti et al., 2003).

Bu tezde Anlamsal Web politikalarının bilgiye güvenli bir biçimde erişim için nasıl kullanılacağına değinilecektir. Bu amaçla bilgiye erişimin ontoloji tabanlı bir yaklaşım ile gerçekleştirildiği Ontoloji Tabanlı Erişim Denetimi modeli geliştirilmiştir. Bu modelde ontolojiler kullanılarak çeşitli politikaların yaratılması gerçekleştirilmektedir.

1.1 Tezin Katkıları

Ontoloji tabanlı bir erişim denetimi modeli gerçekleştirimini hedef alan bu tezin katkıları şu şekilde sıralanabilir:

1. **Ontoloji Tabanlı Erişim Denetimi:** Bu tez çalışmasında, Anlamsal Web teknolojileri kullanılarak temel politika kavramlarını içeren bir Ontoloji Tabanlı Erişim Denetim Modeli (Ontology Based Access Control - OBAC) ortaya konmuştur. Bu modelde bilgiye erişim anlamsal tabanlı bir yaklaşım ile gerçekleştirilmektedir. OBAC modelinde politika nesneleri olarak izin, yasak, zorunluluk ve özel izin; konuşma edimleri olarak yetki aktarma, yetki geri alımı, iptal ve istek tanımlanmaktadır. OBAC, güven ve gizlilik kavramlarından bağımsız bir modeldir.

2. **Üstveri Tanımlanması:** Kaynakçada yer alan modellerde erişilecek kaynak ve bu kaynağa erişecek varlık ile ilgili olarak herhangi bir üstveri bilgisi bulunmamaktadır. OBAC modelinde ise erişilecek kaynak ve bu kaynağa erişecek varlık ile ilgili

üstveriler oluşturulmakta, politikalar da bu üstveriler temel alınarak yaratılmaktadır. Politikaların bir parçası olan üçlüler modelde; kaynağa erişecek varlık özne, kaynağın kendisi nesne ve politika nesneleri yüklem olarak temsil edilmektedir. OBAC modelinde özne, nesne ve politika nesnelerinin yer aldığı politika ontolojileri ayrı ayrı yaratılmaktadır.

3. **Politika Tanımlamalarında Profil Kullanımı:** Roller, sıradüzenler, gruplar ve özellikleri temel alan erişim denetim modellerinden farklı olarak OBAC modelinin politikaların tanımlanmasında profilleri temel alan bir yaklaşımı vardır.

4. **Kişiselleştirmede Politikanın Kullanılması:** Modelin daha iyi bir kişiselleştirme için kullanıcı gereksinimlerine yönelik olarak davranabilmesi amaçlanmıştır. Bu nedenle etki alanı, profiller ve bu profillerin seçenekleri üzerine politika kuralları tanımlanmaktadır. Kaynakçada yer alan erişim denetim modellerinden farklı olarak kişiselleştirme kavramının erişim denetimine katılması bu çalışmanın katkılarından bir diğerini oluşturmaktadır.

1.2 Tezden Çıkan Yayınlar

Tez süresince yapılan çalışmalar sonucunda ortaya çıkan bilimsel yayınlar aşağıda listelenmektedir.

1. **Can, Ö., Ünalir, M. O.,** 2006, Distributed Policy Management in Semantic Web, International Semantic Web Conference (ISWC 2006), Atlanta, Georgia, LNCS 4273, 970-971p.

   Bu bildiri, 5. Uluslararası Anlamsal Web Konferansı'nın Doktora Seminer programı altında poster bildiri olarak sunulmuştur. Bildirinin içeriği, tez çalışmasının amacından ve gerçekleştirilecek işlemlerden oluşmaktadır.

2. **Tapucu, D., Can, Ö., Bursa, O., Ünalır, M. O.,** 2008, Metamodeling Approach to Preference Management in the Semantic Web, 4th Multidisciplinary Workshop on Advances in Preference Handling (M-PREF 2008), AAAI-08, Chicago, USA, July 13-14, 116p.

   Bu bildiri, 23. Yapay Zeka AAAI Konferansı'nın 4. Çok Disiplinli Seçenek İşleme'de Gelişmeler Çalıştayı programı altında sunulmuştur. Bu bildiride, seçenek yönetimi için geliştirilen üst seçenek modeli yaklaşımından ve oluşturulan seçenek ontolojilerinden bahsedilmektedir.

3. **Can, Ö., Ünalır, M. O.,** 2008, Anlamsal Web'de Ontoloji Tabanlı Politikaların Kullanımı, ELECO 2008, Bursa, 26-30 Kasım, 267-271s.

   Bu bildiri, Elektrik - Elektronik ve Bilgisayar Mühendisliği Sempozyumu'nda sunulmuştur. Bildiri, bilgiye erişim için

ontoloji tabanlı bir yaklaşım kullanıldığını ve erişim düzenekleri ile ilgili bilginin modellenmesi için oluşturulan ontolojileri anlatmaktadır.

1.3 Tezin Kapsamı

Belgenin içeriği şu biçimdedir: ikinci bölümde Anlamsal Web ve teknolojileri anlatılmakta, üçüncü bölümde erişim denetim yöntemlerine değinilmekte, dördüncü bölümde politika kavramı ve terimleri tanımlanmakta, ayrıca kaynakçada yer alan politika dili çalışmaları özetlenmekte, beşinci bölümde dağıtık politika yönetimi yaklaşımından söz edilmekte, altıncı bölümde Ontoloji Tabanlı Erişim Denetimi modeli açıklanmakta, yedinci bölümde bir durum çalışması ile yapılan çalışmalar anlatılmakta, sekizinci bölümde gerçekleştirilen uygulamalar açıklanmakta, son olarak sonuçlar irdelenmekte ve gelecekte yapılacak çalışmalar özetlenmektedir.

2. ANLAMSAL WEB

World Wide Web insanların birbiri ile olan iletişim şeklinin ve iş yaşamının yönünün değişmesine neden olmuştur. Böylelikle, gelişen dünyanın bilgi toplumuna dönüşmesini sağlamıştır. Bu gelişme ayrıca kişilerin bilgisayarlar konusundaki düşüncelerini de değiştirmiştir. Genel olarak hesaplama, bilgi işlem, veritabanı ve ofis işlemlerinde kullanıldıkları düşünülen bilgisayarlar bugün bilgiye erişimin giriş noktası durumuna gelmiştir.

Bugünkü örgünün içeriği insanların kullanımı için uygundur. Kişiler bilgi arayabilir, diğer kişiler ile iletişime geçebilir ve çevrimiçi alışveriş yerlerini gezip buralardan alışveriş yapabilirler. Bu etkinlikler yazılım araçları tarafından tam olarak desteklenmemektedir (Antoniou and Harmelen, 2004). En önemli araçlar arama motorlarıdır. Arama motorları olmadan örgünün başarısından söz edilemez (Antoniou and Harmelen, 2004). Günümüz örgüsünde en temel arama motorları olan Google, Yahoo ve AltaVista anahtar sözcük tabanlı arama motorlarıdır.

Arama motoru ne kadar başarılı olursa olsun kullanıcı ilgili belgelere göz atmalı ve istediği bilgiyi seçip çıkartmalıdır. Böylelikle bilginin bulunup getirilmesi çok zaman alan bir etkinlik olmaktadır. Günümüz örgüsünün temel problemi örgü içeriğinin makine erişebilir (*machine accessible*) olmaması nedeni ile anlamsallıktan yoksun olmasıdır. Örgü kullanıcılarına daha iyi destek verebilmek için, örgü bilgisi makinelerce anlaşılabilir (*machine understandable*) olmalıdır. Bu

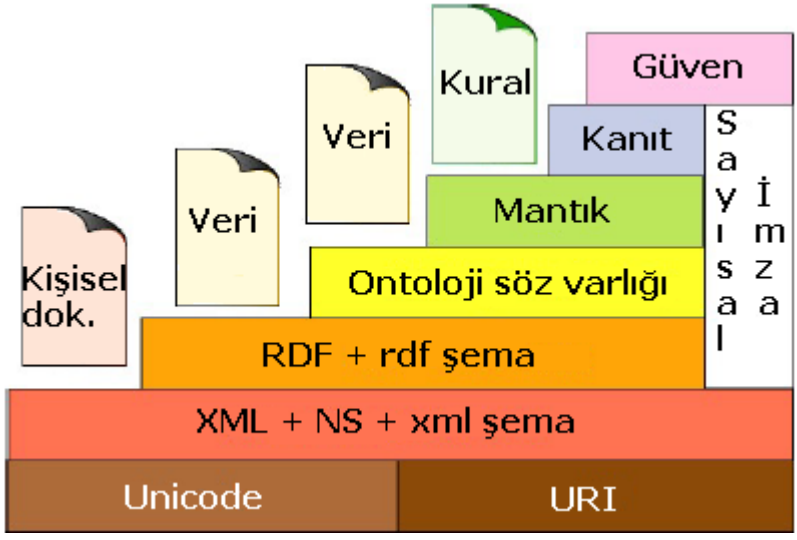
da günümüz örgüsünün Anlamsal Web olarak değiştirilmesidir. Burada önemli olan Anlamsal Web'in günümüz örgüsüne paralel bir şekilde gelişen yeni bir küresel bilgi paylaşımı ortamı olmak yerine aşamalı olarak varolan örgüden geliştirilecek olmasıdır (Antoniou and Harmelen, 2004).

Anlamsal Web, W3C (World Wide Web Consortium) tarafından örgü için uluslararası standart bir gövde olarak geliştirilmiştir. Anlamsal Web girişimini ilk başlatan kişi 1980'li yıllarda WWW'yi geliştiren Tim Berners-Lee'dir. Tim Berners-Lee, Anlamsal Web'de bilginin anlamının günümüz örgüsünde olduğundan daha önemli bir rolde olmasını beklemektedir (Antoniou and Harmelen, 2004).

Şekil 2.1'de Tim Berners-Lee tarafından önerilen Anlamsal Web katmanlı yapısı görülmektedir. En alt katmanda yer alan *XML*[1] *(eXtensible Markup Language)*, kullanıcı tarafından tanımlanmış söz varlığı kullanılarak yapısal örgü belgeleri yazılmasını sağlayan bir dildir. *RDF*, örgü kaynakları ile ilgili olarak yalın ifadeler yazılan varlık-ilişki modeline benzer yalın bir veri modelidir. RDF, XML' e dayanmayan ancak XML tabanlı bir söz dizimi olan bir veri modelidir. Bu nedenle XML katmanının üstünde yer almaktadır. *RDF Şema*, örgü nesnelerini sıradüzen içerisine düzenleyen modelleme ilkelleri sağlamaktadır. Sınıflar ve özellikler, alt sınıf ve alt özellik ilişkileri, etki alanı ve erim kısıtlamaları temel ilkellerdir. RDF Şema, RDF tabanlıdır ve ontoloji yazımı için bir ilkel dil olarak görülebilir. Ancak, RDF Şemayı genişleten

[1] XML, http://www.w3.org/XML/ .

ve örgü nesneleri arasında karmaşık ilişkilerin tanımlanmasına izin verecek daha güçlü *ontoloji dillerine* gereksinim vardır. *Mantık* katmanı, ontoloji dilini güçlendirmek ve uygulamaya özel bildirim deyimi bilgisinin yazımına izin vermek için kullanılmaktadır. *Kanıt* katmanı, hem tümdengelimli işlemleri hem de örgü dillerinde kanıtların temsil edilmesini ve kanıt onaylanmasını içermektedir. Son olarak *Güven* katmanı, sayısal imzaların ve güvenilir etmenler, oranlar, sertifika kuruluşları ve tüketicilerin önerilerini temel alan diğer bilgi türlerinin kullanımı ile ortaya çıkmaktadır. Güven, Anlamsal Web katmanlı yapısının en üstünde yer alması nedeniyle çok önemli bir kavramdır. Örgünün kullanılabilirliği, kullanıcılar işlemlerinde güvencede olduklarında ve sağlanan bilginin niteliğine bağlı olarak elde edilecektir (Antoniou and Harmelen, 2004).



Şekil 2.1 Anlamsal Web katmanlı yapısı.

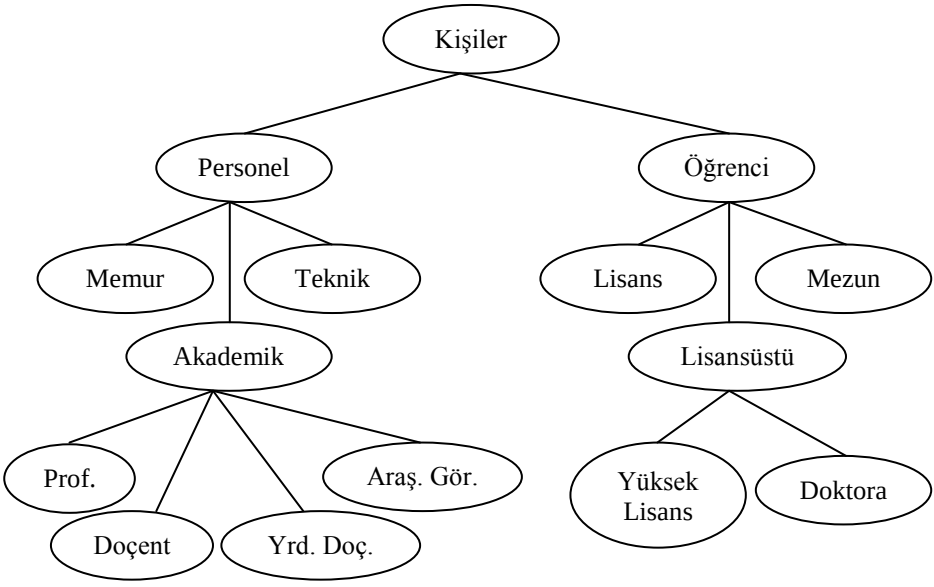
2.1 Ontoloji Dilleri

Ontoloji kelimesi felsefeden gelmektedir. Kelimenin Yunan dilindeki özgün biçimi Οντολογια'dır ve felsefenin bir alt alanı olan varlık biliminin adıdır (Antoniou and Harmelen, 2004). Genel olarak var olan şeylerin türlerini ve onların nasıl tanımlanacağından söz eder. Yaklaşık 15 yıl önce ontoloji kelimesi bilgisayar bilimleri alanında da kullanılmaya başlanmış ve özgün anlamından farklı olarak teknik bir anlam kazanmıştır.

Thomas R. Gruber tarafından yapılmış olan, daha sonra Rudi Studer tarafından yenilenmiş olan tanımda ontoloji kavramsallaştırmanın açık ve biçimsel bir belirtimidir (Gruber, 1993). Bilginin temsilinde ontolojiler kullanılmaktadır. Ontolojiler nesneler, kavramlar ve ilişkiler tanımlayarak belirli bir etki alanına ilişkin bilginin modellenmesini sağlamaktadırlar. Böylece sistemler arasında verilerin değiş tokuşu için standart kavramsal söz varlıkları belirtilmekte, bilgi yeniden kullanılabilmekte, sorgulamaların yanıtlanması için sunular sağlanmakta ve çok çeşitli dizgeler arasında birlikte işlerliği kolaylaştıran sunular yapılmaktadır. Örneğin, üniversite etki alanı ile ilgili bir ontolojide personel, öğrenciler, disiplinler, dersler ve sınıf bilgileri ontolojinin bazı önemli kavramlarındandır.

Terimler etki alanındaki önemli kavramlarından (nesnelerin sınıflarını) söz etmektedir. Örneğin; profesörler, öğrenciler, bölümler ve dersler üniversite etki alanındaki terimlerdendir. İlişkiler, sınıf

sıradüzenlerini içermektedir. Bir sıradüzen, eğer bir *C* sınıfı başka bir *C'* sınıfının alt sınıfı ise, *C*'nin her bir nesnesinin ayrıca *C'* sınıfında yer aldığını belirtmektedir. Örneğin, araştırma görevlileri personeldir dendiğinde bu üniversite etki alanındaki alt sınıf ilişkisini (*subClassOf*) göstermektedir. Şekil 2.2'de üniversite etki alanı sıradüzeni yer almaktadır.



Şekil 2.2 Üniversite etki alanı ontolojisi sıradüzeni.

İlişkiler dışında ontolojiler aşağıdaki bilgileri içerebilirler:

- Özellikler
- Kısıtlamalar
- Ayrık deyimler

- Nesneler arasında ilişkilerin tanımlanması

Örgü bağlamında, ontolojiler bir etki alanının ortak bir anlamından söz etmektedir. Böyle bir ortak anlam, terim bilimindeki farklılıkların üstesinden gelmek için gereklidir. Örneğin bir uygulamada geçen il kodu başka bir uygulamada alan kodu olarak geçebilir. Başka bir sorun aynı terimin her iki uygulamada farklı anlamlarla söylenmesidir. Örneğin Bilgisayar Bilimleri bir uygulama için ders adını gösterirken diğer bir uygulamada üniversitedeki bölümlerden birini gösterebilir. Bu tür farklılıkların üstesinden gelmek için belirli bir terim bilimini ortak bir ontolojiye eşlemek ya da ontolojiler arasında dolaysız eşlemeler tanımlamak gerekmektedir. Her iki durumda da görülmektedir ki, ontolojiler anlamsal birlikte işlerliği desteklemektedir.

İlerleyen alt bölümlerde XML, XML Şema, RDF, RDF Şema ve OWL ontoloji dilinden söz edilecektir.

2.1.1 XML ve XML Şema

XML (eXtensible Markup Language), yapısal belgeler için bir sözdizimi sağlamaktadır ve belirli bir etki alanında anlamsal olarak zengin biçimleme dilleri yaratmak için kullanılan sözdizimi kuralları kümesidir (Daconta et al., 2003). Başka bir deyişle XML, yeni diller yaratmak için kullanılmaktadır.

XML, biçimleme tanımlamak için evrensel bir üstdildir ve

uygulamadan bağımsız belgeler ve veri yaratmaktadır. Ancak, XML verinin anlamsallığı üzerine herhangi bir şey sağlamamaktadır. Belgelerin anlamları arasında herhangi bir anlamsal kısıt yoktur.

XML, HTML gibi biçim imi yapılarından oluşmaktadır. Bir kitap ile ilgili bilgileri içeren örnek bir XML temsili aşağıdaki gibidir:

```
<book>
     <title>
          Ontoloji Tabanlı Erişim Denetimi
     </title>
     <author>Özgü Can</author>
     <publisher>Ege Universitesi</publisher>
     <year>2009</year>
     <ISBN>010071978</ISBN>
</book>
```

Bir XML belgesi eğer iyi biçimlendirilmiş, yapısal bilgiyi kullanıyor ve bu yapısal bilgiye uyuyor ise geçerlidir.

XML Şema, XML belgelerinin yapısını tanımlayan daha zengin bir dildir. Geçerli XML belgelerini belirli bir söz varlığına ve sıradüzensel yapıya kısıtlamayı sağlamaktadır (Daconta et al., 2003). XML Şema tanımlamaları XML belgeleridir ve XML belgelerini türdeş olmayan söz varlığı ile birleştiren ad boşluğu düzeneği sağlamaktadır.

XML Şemanın sözdizimi XML'i temel almaktadır. XML Şema'nın aşağıdaki açılış biçim imi bulunmaktadır:

```
<xsd:schema
     xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
version="1.0">
```

2.1.2 RDF ve RDF Şema

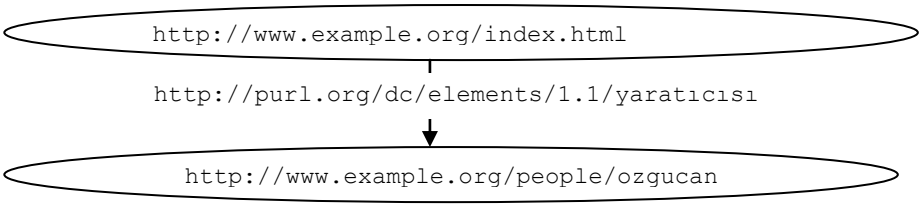
RDF (Resource Description Framework), nesneler ("kaynaklar" – "r*esources*") ve bunlar arasındaki ilişkiler için bir veri modelidir. RDF, bu veri modeli için basit bir anlamsallık sağlar ve bu veri modelleri XML sözdizimi ile gösterilebilir. Aşağıda kişi bilgilerini belirten bir RDF örneği verilmiştir:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
      <contact:Person
            rdf:about="http://www.w3.org/People/OC/contact#me">
          <contact:fullName>Özgü Can</contact:fullName>
          <contact:mailbox
           rdf:resource="mailto:ozgu.can@ege.edu.tr"/>
          <contact:personalTitle>Research
           Assistant</contact:personalTitle>
      </contact:Person>

</rdf:RDF>
```

RDF, *deyim (statement)* olarak adlandırılan nesne-öznitelik-değer (*object-attribute-value*) üçlüsünden oluşmaktadır. Nesne-öznitelik-değer üçlüsü özne-yüklem-nesne üçlüsü olarak adlandırılmaktadır. Aşağıda verilmiş ve Şekil 2.3'de de gösterilmiş olan örnek cümleye göre;

```
http://www.example.org/index.html dosyasının yaratıcısı Özgü
Can'dır.
```



Şekil 2.3 RDF üçlüsü örneği.

- Özne: `http://www.example.org/index.html`
- Yüklem: `yaratıcısı`
- Nesne: `Özgü Can`'dır.

RDF Şema (RDFS), RDF kaynaklarının özelliklerini ve sınıflarını anlamsallıkla tanımlayan söz varlığı tanımlama dilidir. RDF etki alanından bağımsızdır. Bu nedenle, belirli bir etki alanı ile ilgili olarak varsayım yapılamaz (Antoniou and Harmelen, 2004). Kullanıcılar kendi terim bilimlerini RDFS ile tanımlayabilirler. RDF Şema ile RDF arasındaki ilişki XML ile XML Şema arasındaki ilişkiye benzememektedir. XML Şema, XML belgelerinin yapısını kısıtlarken, RDF Şema RDF veri modelinde kullanılan söz varlığını tanımlamaktadır. RDFS'te söz varlığı tanımlanabilir, hangi tür nesnelere hangi özelliklerin uygulanacağı ve hangi değerleri alacakları belirtilebilir ve nesneler arasındaki ilişkiler tanımlanabilir (Antoniou and Harmelen, 2004). RDF/RDFS, belirli etki alanlarının modellenmesini sağlamaktadır.

Aşağıda akademik ve personel sınıf tanımlamalarının yapıldığı bir RDFS örneği verilmiştir:

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

      <rdfs:Class rdf:ID="akademikPersonel">
            <rdfs:comment>
                  Akademik Personel sınıfı
            </rdfs:comment>
            <rdfs:subClassOf rdf:resource="#personel"/>
      </rdfs:Class>

      <rdfs:Class rdf:ID="personel">
            <rdfs:comment>Personel sınıfı</rdfs:comment>
      </rdfs:Class>
</rdf:RDF>
```

2.1.3 OWL

RDF ve RDF Şema'nın betimleme gücü sınırlıdır. W3C'nin Anlamsal Web için belirlediği belirgin kullanım durum çalışmaları RDF ve RDF Şema'nın sunduklarından daha fazla betimleme gücüne gereksinim duymaktadır. OWL (Web Ontology Language), W3C tarafından Anlamsal Web için benimsenmiş standart bir ontoloji dilidir. OWL, DAML+OIL dili başlangıç noktası alınarak geliştirilmiştir.

OWL, sınıflar arasında ilişkilerin tanımlanması, eleman sayısı, eşitlik, özelliklerin zengin bir biçimde sınıflandırılması, özelliklerin nitelikleri ve listelenmiş sınıflar gibi özelliklerin ve sınıfların tanımlanması için zengin bir söz varlığı tanımlama dilidir. W3C tarafından üç farklı OWL altdili tanımlanmıştır:

- ***OWL Full:*** Dilin tamamı OWL Full olarak adlandırılmaktadır. Bütün OWL dili ilkellerini kullanmaktadır. RDF ile sözdizimsel ve anlamsal olarak uyumludur. Geçerli bir RDF belgesi ayrıca geçerli bir OWL Full belgesidir, geçerli bir RDF/RDF Şema kararı (*conclusion*) da ayrıca geçerli bir OWL Full kararıdır.
- ***OWL DL:*** OWL DL (Description Logic), OWL Full'un bir alt dilidir. Etkin bir çıkarsama desteği sağlamaktadır. Ancak RDF ile tam uyum OWL DL'de yitirilmektedir. Bir RDF belgesinin geçerli bir OWL DL belgesi olabilmesi için bazı açılardan genişletilmesi bazı açılardan ise kısıtlandırılması gerekmektedir. Geçerli her bir OWL DL belgesi geçerli bir RDF belgesidir.
- ***OWL Lite:*** Kullanıcıların anlamasının kolay olduğu, araç geliştiriciler için de uygulamanın kolay geliştirilebilir olduğu bir dildir. OWL Lite, kullanıcıların temel gereksinimi olan sınıflandırma sıradüzenini ve basit kısıtları desteklemektedir. OWL DL'den daha az karmaşıklığa sahiptir.

Aşağıda bazı örnek sınıf tanımlamalarının yer aldığı OWL ontoloji parçası verilmiştir:

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl ="http://www.w3.org/2002/07/owl#">
     <owl:Ontology rdf:about="xml:base"/>

     <owl:Class rdf:ID="hayvan">
          <rdfs:comment>Hayvan sınıfı.</rdfs:comment>
     </owl:Class>
```

```
    <owl:Class rdf:ID="bitki">
        <rdfs:comment>
            Bitki sınıfı hayvan sınıfı ile ayrıktır.
        </rdfs:comment>
        <owl:disjointWith rdf:resource="#hayvan"/>
    </owl:Class>

    <owl:Class rdf:ID="ağaç">
        <rdfs:comment>Ağaçlar bitki türüdür.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#bitki"/>
    </owl:Class>
</rdf:RDF>
```

2.2 Kurallar ve Kural Dilleri

RDF, XML üzerine; OWL ise RDF üzerine yapılandırılmıştır. Alt sınıf ilişkileri RDF ile, ek ilişkiler ise OWL ile gösterilebilir. Ancak, çıkarsama işlemi OWL ile bile sınırlıdır. Bu nedenle, makinelerin anlayabilmesi ve çıkarsama yapabilmesi için kurallar ve kurallar için biçimleme dili tanımlanmalıdır (Thuraisingham, 2007). Kurallar, uyulması gereken kısıtları tanımlamaktadır.

Kurallar aşağıdaki biçimde olmaktadır:

$B_1, B_2, \ldots, B_n \rightarrow A$

Eğer B_1 ve B_2 ve ..... ve B_n doğru ise o zaman A'da doğrudur.

Burada *A*, kuralın ardılı; $B_1, B_2, \ldots, B_n$ kuralın öncülleridir. { $B_1, B_2, \ldots, B_n$ } kümesi kuralın gövdesi olarak da adlandırılır.

Örneğin;

Anne(*Selin,Can*) → Aile (*Selin,Can*)

Eğer Selin Can'ın annesi ise, o zaman Selin Can'ın ailesidir.

çalışır(*Can,Bornova*), oturur(*Can,Alsancak*), konum (*Alsancak, İzmir*), konum(*Bornova, İzmir*) → yaşar(*Can, İzmir*)

Eğer Can Bornova'da çalışıyor, Alsancak'ta oturuyor ve bu iki yer İzmir'de bulunuyorsa o zaman Can İzmir'de yaşıyordur.

sürekliAlıcı(*X*), yaş(*X*) > 60 → indirim(*X*)

Eğer alıcı sürekli bir alıcı ve yaşı 60'ın üzerinde ise o zaman bu alıcı indirim kazanmaktadır.

Makinelerin kuralları anlaması için kural biçimleme dillerine gereksinim vardır. RuleML Anlamsal Web için geliştirilmiş olan bir kural biçimleme dilidir. RuleML, XML ve RDF'in özelliklerini birleştiren diğer bir veri modeli sunmaktadır. Aşağıda Aile(*X*) için bir RuleML örneği görülmektedir.

```
<fact>
      <atom>
            <predicate>Aile</predicate>
            <term>
                  <const>X</const>
            </term>
      </atom>
</fact>
```

Bir diğer kural dili olan Anlamsal Web Kural Dili (Semantic Web Rule Language - SWRL) OWL'ın OWL DL ve OWL Lite alt dillerinin

Unary/Binary Datalog RuleML alt dilleri ile bir birleşimidir (Clemente et al., 2005). SWRL, RuleML tabanlı XML sözdizimi ve OWL XML Sunum sözdizimi ile tanımlanmaktadır. Aşağıda "Eğer *X2*, *X1*'in ailesi ise, *X2* ile *X3* kardeş ise ve *X3* erkek ise o zaman *X3*, *X1*'in amcasıdır." kuralı SWRL ile tanımlanmıştır (Horrocks et al., 2004).

```
hasParent(X1,X2), hasSibling(X2, X3), hasSex(X3, male) → hasUncle(X1, X3)
```

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#SWRL Örnek"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSibling">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSex">
      <ruleml:var>x3</ruleml:var>
      <owlx:Individual owlx:name="#male" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

3. ERİŞİM DENETİMİ

Erişim denetimi, insanoğlu değerli varlıklarını korumaya başladığından beri var olan bir kavramdır. Bu nedenle günlük yaşamda geçitler, kilitler ve güvenlik görevlileri gibi dizgeler varlıklara erişimin denetiminde kullanılmaktadır. Günümüz bilgi teknolojilerinde de en temel ve en yaygın güvenlik düzeneği olan erişim denetimi kullanıcıların kaynaklara erişimi ile ilgili olarak "kimin ne yapacağına" karar vermektedir. Erişim denetimi farklı biçimlerde olabilir. Kullanıcının bir kaynağı kullanma iznini belirlemenin yanısıra kaynağın ne zaman ve nasıl kullanılacağını da sınırlayabilir. Örneğin, kullanıcının kaynağa sadece belirli bir zaman dilimi içinde erişmesi tanımlanabilir. Kullanıcı çok kullanıcılı bir dizgeye her bağlandığında erişim denetimi yürütülmektedir.

Erişim denetimi üç temel grupta ortaya çıkmaktadır:

- İsteğe bağlı
- Zorunlu
- Rol tabanlı

Bu bölümde sırası ile bu erişim denetim yöntemleri incelenmektedir.

3.1 İsteğe Bağlı Erişim Denetimi (Discretionary Access Control - DAC)

İsteğe bağlı erişim denetimi (Discretionary Access Control - DAC) kullanıcı merkezlidir ve her bir dizge kaynağı bir ya da daha fazla varlığın iyeliğine atanmıştır. Sistem kullanıcılarının, diğer kullanıcılar tarafından kendi denetimleri altında olan nesnelere erişimine izin vermesine ya da vermemesine olanak vermektedir. Erişim denetiminde kullanılan kaynak ve iyesi kavramı en genel ve gerçek dünyaya uyan bir modeldir. Bir kaynağın iyesi kimin kaynağa erişebileceği ve hangi işlemleri gerçekleştirebileceği ile ilgili tüm kararların iyesidir. İsteğe bağlı erişim denetimi kullanıcıların dizge kaynaklarını denetleme kavramı üzerine yoğunlaşmaktadır. Gerçekleştirilmiş bir çok politikanın bir şekilde ilişkilendiği isteğe bağlı erişim denetimi yaygın bir biçimde benimsenmektedir.

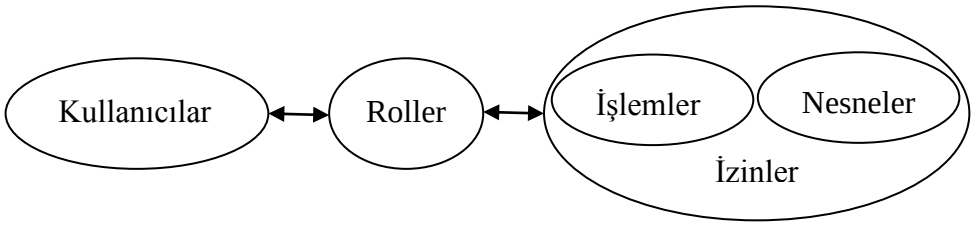
Basitlik, esneklik ve uygulamadaki kolaylığı isteğe bağlı erişim denetiminin üstünlükleridir (Benantar, 2006). Ancak, bilginin akışını ilgilendiren herhangi bir biçimsel güven sağlamaması isteğe bağlı erişim denetiminin sakıncalarındandır. İsteğe bağlı politikalarda erişim yetkilerinin dağıtılması denetimsizdir ve tüm sistemler için öngörülmesi zordur (Benantar, 2006).

3.2 Zorunlu Erişim Denetimi (Mandatory Access Control - MAC)

Zorunlu erişim denetiminde (Mandatory Access Control - MAC) bir kaynağın erişim yetkileri, isteğe bağlı erişim denetiminden farklı olarak, kaynağın iyesi tarafından değil sistem tarafından belirlenmektedir. Zorunlu erişim denetimi kaynak ve iyesi kavramını kullanmamaktadır (Benantar, 2006). Veriye erişim yönetimsel yordamlar ile önceden tanımlanmakta ve daha sonrasında da değişmez kalmaktadır. Sistem varlıklarının veriye erişimin dağıtımına yönelik hiçbir denetimi yoktur. Bunun yerine; erişim özellikleri, güvenilir bir yönetici tarafından, her bir kaynaktaki veriye hangi kullanıcının nasıl erişeceğini belirten bir kural olarak yetkilendirilmektedir. Bir kaynağa erişmek için kullanıcının güvenlik yetkilerini tam olarak sağlaması gerekmektedir.

3.3 Rol Tabanlı Erişim Denetimi

Rol Tabanlı Erişim Denetimi (Role Based Access Control - RBAC) DAC ve MAC politikalarına seçenek olarak ortaya çıkmıştır. RBAC, erişim denetimini eylemde bulunan öznenin rollerine göre düzenlemektedir. RBAC modelinde güvenlik politikasının izinleri kullanıcıya değil rollere verilmektedir. Şekil 3.1 RBAC ilişkilerini göstermektedir (Ferraiolo et al., 2007).



Şekil 3.1 RBAC ilişkileri.

Kullanıcı, izinlerini dizgede tanımlı olan rollerine göre elde etmektedir. Böylece, kullanıcı rolleri ile ilişkili olan bütün izinlerini kalıt almaktadır ve bu rol sıradüzeni ayrıca politikaların tanımlanmasını da basitleştirmektedir (Abou El Kalam et al., 2003; Cuppens and Miège, 2003). Örneğin, bir hastane sistemi düşünüldüğünde hastane ağına erişebilecek roller: `doktor`, `hemşire`, `laboratuar teknikeri`, `memur` ve `yönetici`; bir otel sisteminde ise varolan roller: `yönetici`, `personel` ve `ziyaretçi` şeklinde tanımlanmaktadır. Eğer kullanıcı `personel` rolünde ise kullanıcının otelin oda ile ilgili dosyalarına erişim yetkisi vardır, `yönetici` rolünde olan kullanıcı ise hem otel oda dosyalarına hem de muhasebe ile ilgili dosyalara erişim yetkisi vardır. Ancak, RBAC modelinde, *kullanıcı-rol* ve *izin-rol* atamaları için yönetimsel işlemlerin gerekli olması, roller ve izinlerin sayısındaki üstel artışın *kullanıcı-rol* ve *izin-rol* atama işlemlerini pahallı bir duruma getirmesi gibi bazı problemler bulunmaktadır (Yuan and Tong, 2005a). RBAC modelinde yer alan bu problemler nedeni ile bu tezde profil tabanlı politika yaklaşımı kullanılmaktadır.

3.3.1 RBAC politika yöntemi

RBAC modelinin altyapısında her bir kullanıcıya bir rol verilmekte, izinler rollere atanmakta, roller nesneler ile ilişkilendirilmekte ve erişim kararları uygulanabilir rollerin üyesi olan kullanıcıları temel almaktadır. Bu modelin temel özelliği erişim denetiminin roller üzerinden gerçekleştirilmesi olduğundan roller izinlerden oluşmakta ve bütün kullanıcılar izinlere sadece ilgili oldukları roller üzerinden iye olmaktadırlar (Ferraiolo et al., 2007). RBAC politika düzenlemesi şu şekildedir (Ferraiolo and Kuhn, 1992):

1. Her bir özne için, etkin rol o öznenin o anda kullandığı roldür:

$$AR^{1}(s: \text{özne}) = \{s \text{ öznesi için etkin } rol\}$$

2. Her özne bir ya da daha fazla rolü uygulamak için yetkilendirilebilir:

$$RA^{2}(s: \text{özne}) = \{s \text{ öznesi için yetkilendirilmiş } roller\}$$

3. Her rol bir ya da daha fazla işlemi gerçekleştirmek için yetkilendirilebilir:

$$TA^{3}(r: rol) = \{r \text{ rolü için yetkilendirilmiş işlemler}\}$$

4. Özneler işlemleri gerçekleştirebilirler. *exec(s, t)* yüklemi eğer

---

[1] Etkin Rol (*Active Role*)

[2] Yetkilendirilmiş Roller (*Authorized Roles*)

[3] Yetkilendirilmiş İşlemler (*Authorized Transactions*)

ve sadece s öznesi t işlemini o anki zamanda yürütürse doğrudur, aksi halde yanlıştır:

$$exec(s\text{: özne}; t\text{: işlem})$$
$$= \{doğru, eğer\ s\ öznesi\ t\ işlemini\ yürütürse\}$$

i. Rol ataması: Bir özne bir işlemi sadece özne bir role atanmış ya da rolü seçmiş ise yürütebilir:

$$\forall s\text{: özne}, t\text{: işlem} \cdot exec(s,t) \Longrightarrow AR(s) \neq \emptyset$$

ii. Rol yetkilendirmesi: Bir öznenin etkin rolü özne için yetkilendirilmelidir:

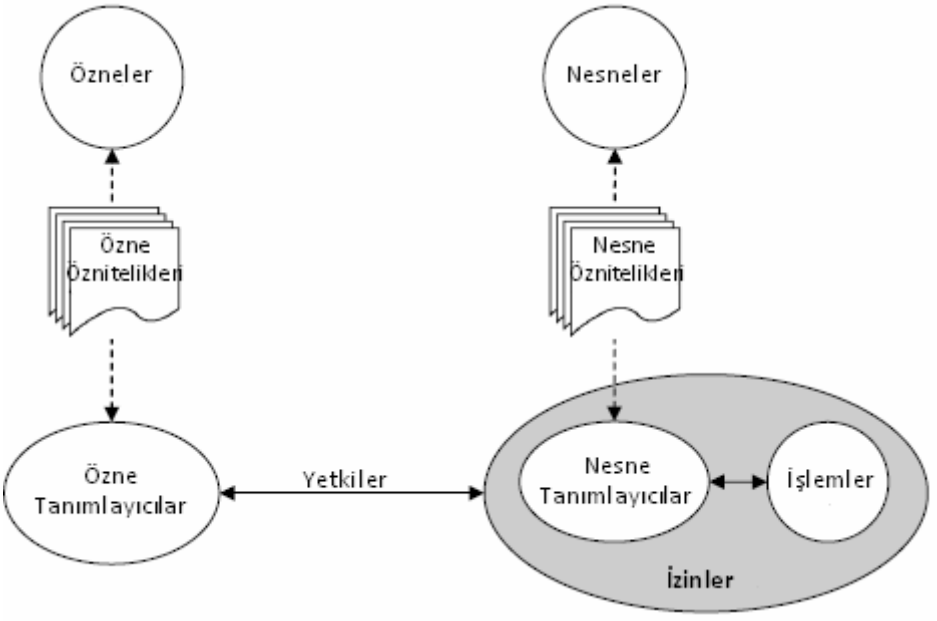
$$\forall s\text{: özne} \cdot AR(s) \subseteq RA(s)$$

iii. İşlem yetkilendirmesi: Bir özne bir işlemi sadece işlem öznenin etkin rolü için yetkilendirilmiş ise yürütebilir:

$$\forall s\text{: özne}, t\text{: işlem} \cdot exec(s,t) \Longrightarrow t \in TA(AR(s))$$

3.4 Öznitelik Tabanlı Erişim Denetimi

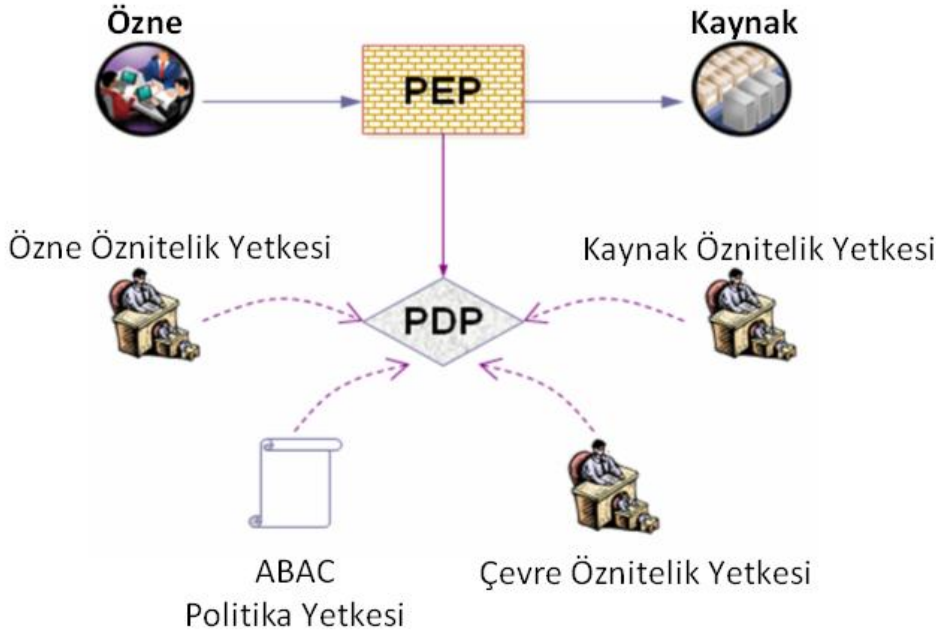
Öznitelik Tabanlı Erişim Denetimi (Attribute Based Access Control - ABAC) özneler ve nesneler arasında izinleri doğrudan tanımlamak yerine, yetkiler için onların özniteliklerini kullanmaktadır. Özneler için öznitelikler durağan olarak öznenin düzenlemedeki rolü, devingen olarak ise yaşı olabilir. Nesneler için öznitelikler ise bir belgenin konusu gibi üst veri özellikleridir. Bu kavram Şekil 3.2'de gösterilmektedir (Priebe et al., 2006).



Şekil 3.2 ABAC modelinin genel görünümü.

ABAC'te öznitelikler üç şekilde gösterilebilir. Bunlar: Özne Öznitelikleri (*Subject Attributes*), Kaynak Öznitelikleri (*Resource Attributes*) ve Çevre Öznitelikleri (*Environment Attrib*utes) (Yuan and Tong, 2005b).

Şekil 3.3'de gösterilen ABAC modeli yapısında yer alan varlıklar Öznitelik Yetkeleri (*Attribute Authorities - AA*), Politika Yürütme Noktası (*Policy Enforcement Point - PEP*), Politika Karar Noktası (*Policy Decision Point - PDP*) ve Politika Yetkesi'dir (*Policy Authority - PA*) (Yuan and Tong, 2005a).



Şekil 3.3 ABAC yetki mimarisi.

Öznitelik Yetkeleri (AA), sırasıyla özneler, kaynaklar ve çevre için özniteliklerin yaratılması ve yönetilmesinden sorumludurlar. Mantıksal bir varlık olarak bir AA, öznitelikleri kendisi saklamaz, fakat öznitelikleri bir varlık ile ilişkilendirmekten sorumludur. Özniteliklerin saptanması ve elde edilmesinde önemli bir rol oynar.

Politika Yürütme Noktası (Policy Enforcement Point - PEP), yetki kararlarını istemekten ve bu kararların yürütülmesinden sorumludur. Erişim denetiminin var olabilmesinde asıl nokta PEP'tir ve bilgiyi isteyen ile bilgiyi sunan arasındaki sunu isteklerini durduracaktır. Şekil 3.3'de PEP tek bir nokta olarak gösterilmiş olsa da fiziksel olarak ağ boyunca

dağıtık olabilir. PEP'in uygulanmasındaki en önemli nokta, korunan varlığa PEP'e uğramadan geçilemeyecek biçimde dizgenin tasarlanmasıdır.

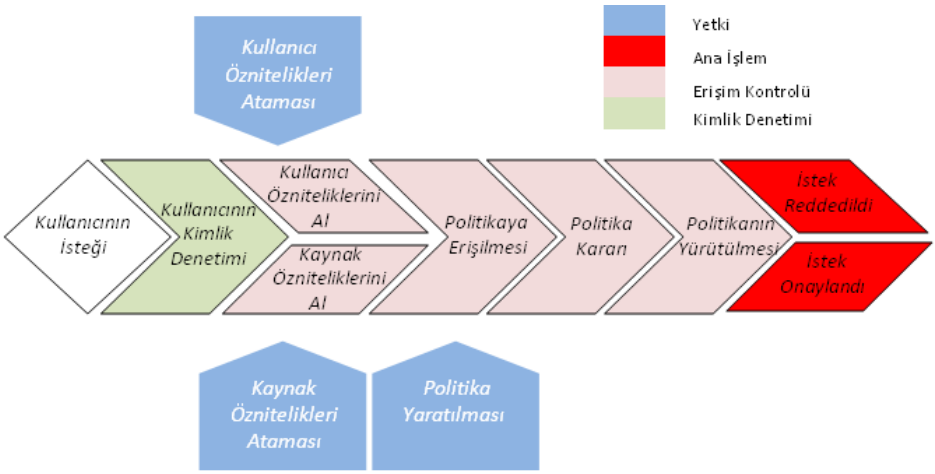
Politika Karar Noktası (Policy Decision Point - PDP), uygulanabilir politikaların değerlendirilmesinden ve yetki kararının (onay ya da red) verilmesinden sorumludur. PDP, aslında bir politika yürütme (*execution*) motorudur. Bir politika istekte yer almayan bir özne, kaynak veya bir çevre özniteliğinden sözediyorsa, öznitelik değer(ler)ini elde edeceği ilgili öznitelik yetkeleri ile iletişim kurar.

Politika Yetkesi (Policy Authority - PA), erişim denetim politikalarını yaratır ve yönetir. Politikalar, kaynaklara erişmek için, karar kurallarından, koşullarından ve diğer kısıtlardan oluşur.

Özne Öznitelikleri, öznenin kimliğini ve niteliklerini tanımlayan özne (kullanıcı, uygulama, işlem) ile ilişkilenmektedir. Örneğin; kimlik, isim, meslek ya da rol bilgisi. *Kaynak Öznitelikleri*, örgü sunusu, dizge işlevi ya da veri gibi bir kaynakla ilişkilenmektedir. Örneğin; Dublin Core[1] üstveri elemanları. Bu tez ile ilşkilendirildiğinde özne öznitelikleri profil, kaynak öznitelikleri ise etki alanı bilgisi tanımlayıcısı olarak belirtilebilir. *Çevre Öznitelikleri* işlemsel, teknik veya durumsal çevreyi ya da bilgi erişiminin gerçekleştiği bağlamı göstermektedir. Örneğin; geçerli tarih/saat bilgisi, geçerli tehlike düzeyi.

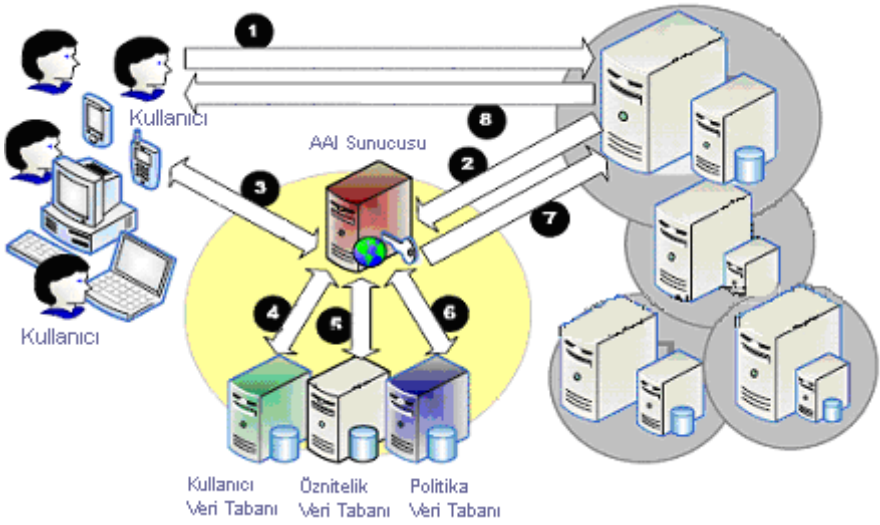
[1] Dublin Core, http://dublincore.org/ .

Özne ve kaynak öznitelikleri kullanılarak kaynaklara erişimin onaylanması işlemi Şekil 3.4'te gösterilmektedir (Priebe et al., 2007). Kullanıcı bir istekte bulunduktan sonra kullanıcının kimlik denetimi gerçekleştirilmektedir. Daha sonra, kullanıcı ve kaynak öznitelikleri alınıp politikaya erişilmektedir. Burada, öznitelikler kimlik ve profil bilgisi içerebilirler. Politika kararından sonra politika yürütülerek isteğin onaylanması ya da reddedilmesi işlemi gerçekleştirilmektedir.



Şekil 3.4 Öznitelik altyapısını kullanan güvenlik sunuları.

Ortadaki bir öznitelik yönetimini kullanan genel bir altyapıda yer alan iletişim basamakları Şekil 3.5'te gösterilmiştir.



Şekil 3.5 ABAC tabanlı merkezi öznitelik yönetimi.

Burada gerçekleşen işlem adımları şu biçimdedir:

1. Kullanıcı, sunu sağlayıcısında erişmek istediği kaynak için istekte bulunur.

2. Sunu sağlayıcısı, güvenlik sunuları için Kimlik Denetimi ve Yetki Altyapı Dizgesi'ni (Authentication and Authorization Infrastructure - AAI) görevlendirdiğinden, erişim denetimi kararı için istek bu sunucuya iletilir.

3. Kullanıcı kimlik denetimi gerçekleştirildikten sonra, AAI, kullanıcı özniteliklerini ortadaki bir veri tabanından alır.

4. 5. ve 6. adımlarda istek ile ilgili kararı vermek için erişim denetimi politikasını kullanır. Daha sonra, politika kararı sunu sağlayıcısına iletilir. Gerekli olması durumunda, sunu sağlayıcısı kullanıcının özniteliklerini günceller. Burada özen gösterilmesi gereken

nokta kullanıcı verisinin sunu sağlayıcısında değil AAI'da saklandığıdır.

3.4.1 ABAC politika yöntemi

ABAC politika üzenlenmesi şu biçimdedir (Yuan and Tong, 2005b):

1. *S*, *R*, ve *E*: sırası ile; özneler, kaynaklar ve çevrelerdir.
2. SA_k ($1 \le k \le K$), RA_m ($1 \le m \le M$), ve EA_n ($1 \le n \le N$): sırası ile özneler, kaynaklar ve çevreler için önceden tanımlanmış özniteliklerdir.
3. *ATTR(s)*, *ATTR(r)*, ve *ATTR(e)*: özne *s*, kaynak *r* ve çevre *e* için öznitelik ilişkilerinin atanmasıdır.

$$ATTR(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_K$$
$$ATTR(r) \subseteq RA_1 \times RA_2 \times \ldots \times RA_M$$
$$ATTR(e) \subseteq EA_1 \times EA_2 \times \ldots \times EA_N$$

4. Tek bir özniteliğe değer atanması için de işlev gösterimi kullanılmaktadır. Örneğin:

$$Role(s) = "Doktor"$$
$$ServiceOwner(r) = "Ege\ Üniversitesi"$$
$$CurrentDate(e) = "12 - 03 - 2007"$$

5. Bir politika kuralı, belirli bir *e* çevresinde, *s* öznesinin *r* kaynağına erişip erişemeyeceğine karar verir. Bu durum *s*, *r* ve *e* özniteliklerinin bir Boole işlevi olarak gösterilir.

$$Rule\ X: can\_access(s, r, e) \leftarrow f(ATTR(s), ATTR(r), ATTR(e))$$

Politika dağarcığında yer alan politika kurallarının uygulanabilirliğinin değerlendirilmesi ile erişim denetimi kararı verilir.

3.4.2 ABAC politika kuralı örnekleri

Aşağıda örnek ABAC politika kuralları verilmiştir:

Doktor rolünde olan kullanıcı ReçeteOnayla sunusuna erişebilir.

$$Rule1: can\_access(s,r,e) \leftarrow Role(s) = "Doktor" \wedge Name(r) = "ReçeteOnayla"$$

Bir kaynak sadece o kaynağın iyeleri tarafından erişilebilir.

$$Rule2: can\_access(s,r,e) \leftarrow UserID(s) = OwnerID(r)$$

Sınıflandırılmış dosyalar eşit ya da daha yüksek yetkileri olan kişiler tarafından erişilebilir.

$$Rule3: can\_access(s,r,e) \leftarrow Clearance(s) \geq Classification(r)$$

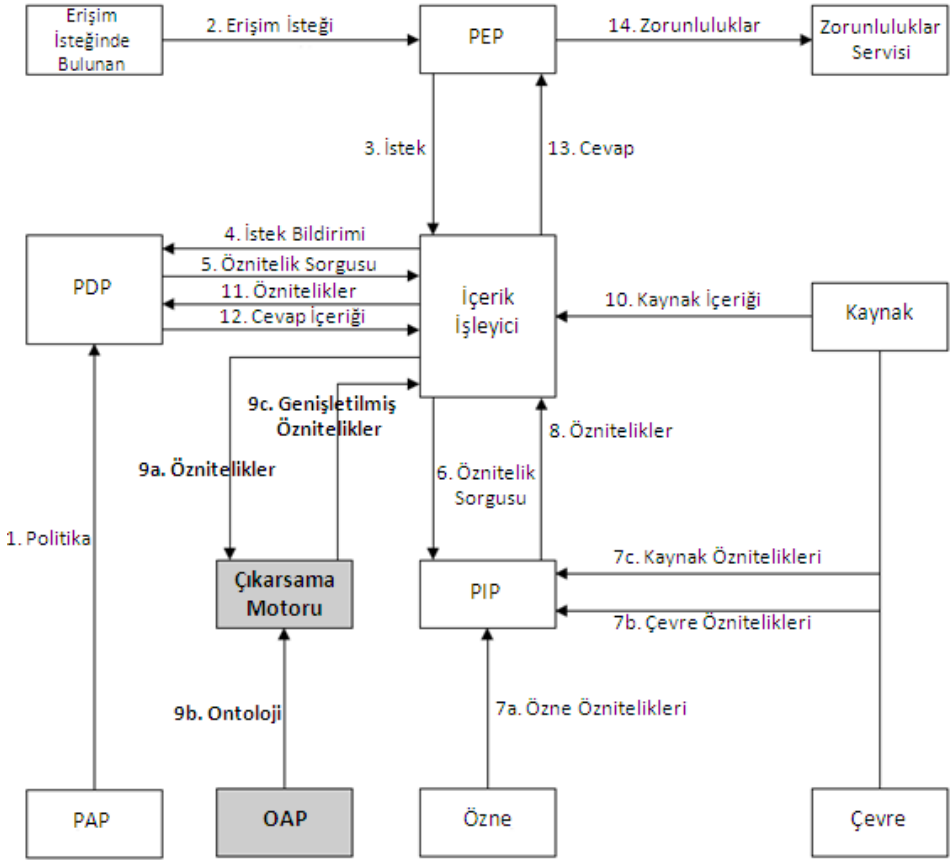
3.4.3 Önerilen ABAC Mimarisi

Öznitelik tabanlı erişim denetiminde, denetlenmesi gereken bütün kullanıcı öznitelikleri ve koşullarının (örneğin; "`age>18`" gibi) politikada belirtilmesi gerekmektedir. Politikaları tanımlarken, sadece dizge tasarımcısı için gerekli olan koşullar uygun olmalıdır (örneğin;

"`fullAge=true`" gibi). Kullanıcının bu özniteliği nasıl sağlayacağı özniteliklerin yönetilmesi ile ilgili bir durumdur. Bu amaçla, Anlamsal Web teknolojileri kullanılarak XACML mimarisi genişletilmiştir (Priebe et al., 2006). XACML, ABAC yapısını destekleyen XML tabanlı politika dilidir. Tezin 4.2.5 kısmında XACML politika dili açıklanmaktadır. Şekil 3.6'da genişletilmiş XACML mimarisi gösterilmektedir. Farklı öznitelikler ve öznitelik koşulları arasındaki eşleme, ontoloji tabanlı çıkarsama motoru ile gerçekleştirilmektedir. (Priebe et al., 2006) çalışması ile genişletilmiş olan XACML mimarisi gri gölgeleme ve kalın çizgiler ile tanımlanmıştır.

Erişim denetimi kararı ve yürütülmesi aşağıdaki adımlar izlenerek gerçekleştirilmektedir (Priebe et al., 2006):

1. Politika Yönetim Noktası (*Policy Administration Point - PAP*), Politika Karar Noktası (*Policy Decision Point - PDP*) için bir XACML politikası yaratır.
2. İstekte bulunan kullanıcı, kaynak isteğini Politika Yürütme Noktası'na (*Policy Enforcement Point - PEP*) iletir.
3. PEP, kullanıcı, kaynak ve çevre özniteliklerini içeren bu isteği içerik yürütücüsüne iletir. Kullanıcının e-posta adresi ile tanımlandığı ve "`age`" özniteliğinin iyesi olduğu varsayılmaktadır.



Şekil 3.6 Genişletilmiş XACML mimarisi.

4. İçerik yürütücüsü bir XACML isteği yaratır ve bu isteği PDP'ye gönderir.

5. PDP'nin daha fazla özne, kaynak ve çevre özniteliklerine gereksinim duyması durumunda bu bilgiler içerik yürütücüsünden istenir. Bu örnekte, PDP "`fullAge`" özelliğine gereksinim duymaktadır.

6. İçerik yürütücüsü bu öznitelikleri Politika Bilgi Noktası'ndan (*Policy Information Point - PIP*) ister.

7. PIP istenilen öznitelikleri, eğer olabilirse, özne, kaynak ve çevreden toplar. Bu durumda, "`fullAge`" özniteliğinin kullanıcıdan alınması olasılığı yoktur.

8. PIP öznitelikleri içerik yürütücüsüne iletir.

9. Bu aşamaya kadar olan işlemler XACML yordamında açıklandığı biçimdedir. Eğer, PDP'den istenen bazı öznitelikler yoksa, Anlamsal Web teknolojileri kullanılarak, PIP tarafından sağlanır. Öznitelikler 9a adımında gösterildiği gibi çıkarsama motoruna iletilir. Çıkarsama motoru, Ontoloji Yönetim Noktası (*Ontology Administration Point - OAP*) tarafından iletilen ontoloji ile öznitelikleri birleştirir. Örneğin, "`fullAge`" özniteliği "`hasDriverLicense`"dan elde edilmektedir. "`age`" özniteliğine bağlı olan "`fullAge`"i tanımlayan SWRL kuralı şu şekildedir:

```
Subject(?x) ∧ age(?x, ?a) ∧ greaterThanOrEqual(?a, 18) ⇒
                        fullAge(?x, true)
```

9b adımında, çıkartılan öznitelikler daha sonra içerik yürütücüsü tarafından `DESCRIBE` komutunu kullanan SPARQL isteği ile sorgulanır. Örneğin:

```
DESCRIBE <mailto:user@example.org>
```

Çıkarsama motoru, 9c adımında, tüm öznitelikleri içerik yürütücüsüne iletir.

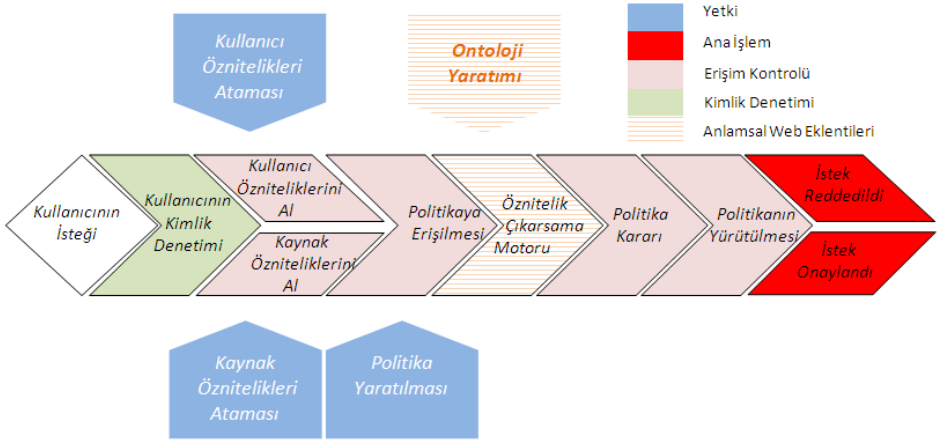
10. Bu adımdan sonra işlemler yine XACML yordamında açıklandığı biçimdedir. İsteğe bağlı olarak, içerik yürütücüsü kaynağın kendisinide isteğe iliştirir.

11. Genişletilmiş istek PDP'ye gönderilir.

12. PDP politikayı değerlendirir ve erişim denetimi kararını içeren yanıt

içeriğini içerik yürütücüsüne geri gönderir.
13. İçerik yürütücüsü, yanıt içeriğini PEP'in yerel biçimine çevirip iletir.
14. PEP olası zorunlulukları karşılar.
Eğer erişim denetimi onaylanırsa, PEP kaynağa erişime izin verir. Tersi durumda erişim isteği reddedilir.

Yukarıda anlatılan mimari yani Şekil 3.4'de gösterilmiş olan sisteme çıkarsama motorunun eklendiği durum Şekil 3.7'de gösterilmektedir. Burada, Şekil 3.4'den farklı olarak, politikaya erişilmesi ile politika kararı arasına öznitelik çıkarsama motoru eklenmektedir.



Şekil 3.7 Öznitelik çıkarsama motoru kullanılarak geliştirilen güvenli sunular.

Çıkarsama motoru, yeni öznitelikler veya öznitelikler arasındaki eşlemeleri çıkarsamaktadır. Böylelikle, açık sistemlerde esnek ve

devingen öznitelik değişimi gerçekleştirimi sağlanmış olacaktır. Kullanıcı bilgisinin anlamsal eşlenmesi nedeniyle, merkezi havuz yönetimine gerek kalmayacaktır (Priebe et al., 2007).

3.5 ABAC ve RBAC Karşılaştırması

ABAC ve RBAC karşılaştırmasını yaparken Çevrimiçi Eğlence Mağazası durum çalışması kullanılacaktır (Yuan and Tong, 2005a). Bu durum çalışmasına göre satıcı kurum, kullanıcılarının yaşını ve filmlerin içerik sınıflarını (rating) temel alan bir erişim denetimi politikası uygulamaktadır. Film sınıfları ve ilgili erişim denetim politikası Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1 Erişim denetimleri.

Film Sınıfı	**İzin Verilen Kullanıcı**
`Yas_18`	`18 yaş ve üzeri`
`Yas_12`	`12 yaş ve üzeri`
`H`	`Herkes`

RBAC modelinde, kullanıcı rolleri göz önüne alındığından bu durum çalışması için üç rol yaratılacaktır: `Yetişkin`, `Genç` ve `Çocuk`. Her kullanıcı sisteme kayıt olurken bu üç rolden birisi ile ilişkilendirilecektir. Filmlerin izlenmesi ile ilgili olarak üç tane izin

yaratılacaktır: "*Yas_18 sınıfındaki filmleri izleyebilir.*", "*Yas_12 sınıfındaki filmleri izleyebilir.*" ve "*H sınıfındaki filmleri izleyebilir.*". `Yetişkin` rolünün bu üç izini varken `Genç` rolü "*Yas_12 sınıfındaki filmleri izleyebilir.*" ve "*H sınıfındaki filmleri izleyebilir.*" izinlerine ve `Çocuk` rolünün ise sadece "*H sınıfındaki filmleri izleyebilir.*" izni vardır. *Kullanıcı-rol* ve *izin-rol* atamaları yöneticinin görevleri arasına girmektedir.

ABAC yaklaşımında ise rollerin tanımlanmasına gerek yoktur. Bir `k` kullanıcısının bir `f` filmine erişme veya filmi izlemeye ilişkin aşağıdaki politika kuralının değerlendirilmesi ile karar verilmektedir:

$$R1: can\_access(k, f, e) \leftarrow (Yas(k) \geq 18) \wedge Sınıf(f) \in \{Yas_{18}, Yas_{12}, H\}) \vee (Yas(k) \geq 12 \wedge Yas(k) < 18 \wedge Sınıf(f) \in \{Yas\_12, H\}) \vee (Yas(k) < 12 \wedge Sınıf(f) \in \{H\})$$

Burada `Yas` ve `Sınıf` sırasıyla özne ve kaynak özniteliklerdir. ABAC modelinin üstünlüğü rollerin tanımlanması ve yönetimi işlemlerini yok etmiş olmasıdır. Böylece *kullanıcı-rol* ve *izin-rol* atamaları için gerekli olan yönetim görevleri de ortadan kalkmaktadır (Yuan and Tong, 2005a).

Yukarıdaki durum çalışmasında Çevrimiçi Eğlence Satıcı Kurumu kullanıcılarının aylık olarak belirli bir ücret ödedikleri dikkate alınarak kullanıcılar ödedikleri miktara göre sınıflandırılmaktadırlar. Böylece

Öncelikli Kullanıcı ve Standart Kullanıcı olmak üzere iki kullanıcı türü oluşmaktadır. Filmler piyasaya çıkma tarihlerine göre Yeni Sürüm ve Eski Sürüm olarak sınıflandırılmaktadırlar. Mağaza, "*Öncelikli kullanıcılar yeni sürümleri izleyebilir.*" şeklinde bir politika tanımlamak istemektedir. Bu durumda RBAC kullanıldığında tanımlanacak kullanıcı rolleri ve izinler Çizelge 3.2 ve Çizelge 3.3'de gösterildiği gibi iki katına çıkmaktadır.

Çizelge 3.2 RBAC rolleri.

RBAC Rolleri
Yetişkin Öncelikli Kullanıcı Rolü
Yetişkin Standart Kullanıcı Rolü
Genç Öncelikli Kullanıcı Rolü
Genç Standart Kullanıcı Rolü
Çocuk Öncelikli Kullanıcı Rolü
Çocuk Standart Kullanıcı Rolü

Çizelge 3.3 RBAC izinleri.

RBAC İzinleri
Yas_18 sınıfındaki yeni sürüm filmleri izleyebilir.
Yas_18 sınıfındaki eski sürüm filmleri izleyebilir.
Yas_12 sınıfındaki yeni sürüm filmleri izleyebilir.
Yas_12 sınıfındaki eski sürüm filmleri izleyebilir.
H sınıfındaki yeni sürüm filmleri izleyebilir.
H sınıfındaki eski sürüm filmleri izleyebilir.

Genel olarak, eğer K tane özne öznitelikleri ve M tane kaynak öznitelikleri mevcut ise her bir öznitelik için alabileceği değerler aralığını Range() belirtmektedir. Bu durumda yaratılacak roller ve izinler sayısı şu şekilde olacaktır (Yuan and Tong, 2005a):

$$\coprod_1^K Range(SA_{\mathrm{k}}) \text{ ve } \coprod_1^M Range(SA_{\mathrm{m}})$$

Bu nedenle, öznitelik sayısı arttıkça rollerin ve izinlerin sayısı üstel olarak artacaktır.

ABAC modelinde ise, bir önceki R1 politika kuralı geçerliliğini korumakta olduğundan sadece ikinci bir R2 kuralının eklenmesi ve bu iki kuralın birleşiminin alınması gerekmektedir.

$$R2: can\_access(k, f, e) \leftarrow (\text{Üyelik}Tipi(k) = '\text{ Öncelikli}') \vee (\text{Üyelik}Tipi(k) = ' Standart' \wedge FilmTipi(f) = 'EskiSürüm')$$

$$R3: can\_access(k, f, e) \leftarrow R1 \wedge R2$$

Bu örneklerde çevre öznitelikleri kullanılmamıştır. Eğer "*Standart kullanıcılar tanıtım dönemlerinde yeni filmleri izleyebilirler.*" şeklinde bir kural tanımlandığında ABAC'te günün tarihi denetlenerek elde edilen bu çevre özniteliği tanıtım dönemi olarak uygulanmaktadır. RBAC modeli çevre özniteliklerini açıkça belirtmemektedir.

3.6 Kurum Tabanlı Erişim Denetimi (OrBAC)

RBAC, DAC ve MAC gibi erişim denetim modellerinin politikaların kurumlarda uygulanmasında yaşanan sorunlar nedeni ile Kurum Tabanlı Erişim Denetimi (Organization Based Access Control - OrBAC) sistemi geliştirilmiştir. Varolan erişim denetim modelleri 3 varlık ile çalışmaktadır: özne (*subject*), eylem (*action*) ve nesne (*object*). Erişimi denetlemek amacı ile bazı öznelerin bazı nesneler üzerinde bazı eylemleri gerçekleştirmelerine izinleri olduğu politikada tanımlamaktadır. OrBAC modeli uygulamadan bağımsız olarak politika yazılabilmesine izin vermektedir. OrBAC sisteminde iki düzey bulunmaktadır (Cuppens and Miège, 2003):

- **Somut (*Concrete*):** özne, eylem, nesne
- **Soyut (*Abstract*):** rol (*role*), etkinlik (*activity*), görüş (*view*)

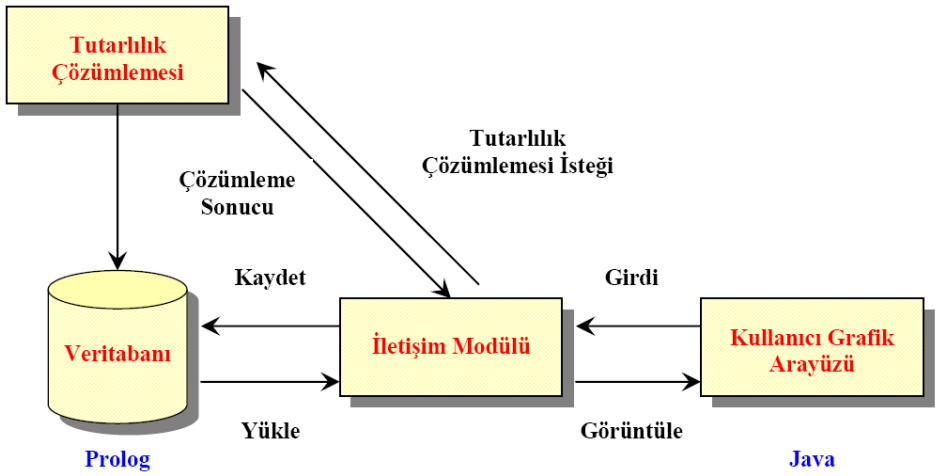
Rol, kuralların uygulandığı özneler kümesidir. Çalışma, kuralların uygulandığı eylemler kümesi, görüş ise kuralların uygulandığı nesnelerin kümesidir.

OrBAC modelinde politikalar devingendir. İzin, yasak, zorunluluk ve öğüt politika nesnelerini içermektedir.

OrBAC modeli politikalarının kolaylıkla yaratılması amacıyla MotOrBAC[1] ana örneği geliştirilmiştir. Şekil 3.8'de görülen MotOrBAC birimleri 4 bölümdür:

i. Politika Tutarlılık Çözümlemesi: Politika çelişkileri saptanıp çözülmektedir.
ii. Verinin Kaydedilmesi: Oluşturulan politikalar saklanmakta ve politikaların yeniden yüklenmesi gerçekleştirilmektedir.
iii. İletişim Modülü: Kullanıcı ile MotOrBAC arasındaki iletişimin gerçekleşmesini sağlamaktadır.
iv. Grafik Arayüzü: Java ile gerçekleştirilmiş olan grafik arayüzü kullanıcılara politikaların yaratılmasında kolaylık sağlamaktadır.

[1] MotOrBAC, http://motorbac.sourceforge.net.



Şekil 3.8 MotOrBAC birimleri

4. POLİTİKA

Politikalar günlük yaşamımızda bir çok alanda karşılaştığımız bir kavramdır. Erişim denetimi, eğitim, hükümet ve sağlık politikaları gibi çeşitli politika kavramları bulunmaktadır. Politika, dizgenin davranış şeklini belirten bir durumdur. Bir karar verme süreci olan politika, hem bir karar destek sistemi hem de bildirimsel deyim (*declarative*) davranış sistemi olarak davranmaktadır (Thuraisingham, 2007). Bildirimsel deyim bir eylemin yapılıp yapılamayacağının bilinmesi durumudur.

Kural, davranışları belirleyen bir durumdur. Politika en basit tanımı ile kurallar kümesidir. Sadece belirtilmiş belgeleri (*credentials*) olan varlıklar işlemleri gerçekleştirebilir. Eğer varlık gereksinimleri karşılıyorsa güvenilir olarak alınmakta ve işlemleri gerçekleştirmesine izin verilmektedir. Bir politika üçlüler (*triple*) biçiminde gösterilir (Quinn et al., 2004):

- **Olay (*Event*):** Politika kuralını tetikler.
- **Koşul (*Condition*):** Politikanın uygulanması için meydana gelmesi gereken koşulların kümesidir.
- **Eylem (*Action*):** Eğer politika uygulanmaya başladıysa yerine getirilmesi gerekenler eylemlerdir.

Aşağıda çeşitli kural örnekleri verilmiştir:

EGE'den mezun bütün öğrenciler yazdır isimli bir örgü sunusuna erişebilir;

```
has(x, right(yazdır, mezunOgrenci(x, 'EGE')))
```

'Ahmet' yazdır işlemini gerçekleştirdikten sonra yeniden renkliyazdır işlemini veya bir kerelik faks işlemini gerçekleştirme yetkesi verilmiştir;

```
has('Ahmet',right(nond(seq(yazdır,iteration(renkliYazdır)),
        once(faks)), lab-uyesi('Ahmet', 'Ege Lab')))
```

Ali'nin eski eşi Ayşe'ye, Ayşe'nin yeniden evlenmesi durumunda, nafaka ödemek zorunda olmadığını belirten kural şu şekilde tanımlanmaktadır;

```
has(Ali,dispensation(nafakaOdemesi,[yenidenEvlenme(Ayşe)]))
```

Varlıkların davranış biçimlerine göre politikalar zaman içerisinde değişebilmektedir. Dizgenin denetlenmesi için politikalar çalışma zamanında güncellenebilmektedir. Politika durağan ya da devingen olabilir. *Durağan politika*, eyleme başlamadan önce herkese verilmiş olan politika; *devingen politika* ise çalışma durumuna göre eylemin ileriki durumlarında değiştirilen politikadır.

4.1 Politika Tanımlama Terimleri

Bu bölümde politika kavramı, bu kavram ile ilişkili diğer kavramlar, politika tanımlamada kullanılan terimler ve bu terimlerin

tanımları yer almaktadır.

Politika (*Policy*): Politika kurallar kümesidir. Bir sunuyu, kimin ve hangi koşullar altında kullanabileceğini, bilginin sunuya nasıl sağlanacağını ve sağlanan bilginin nasıl kullanılacağını belirtir (Kagal et al., 2004).

Kimlik Denetimi (*Authentication*): Alıcının, veriyi gönderen kişinin belirttiği kişi olduğunu doğrulamasıdır. Bir başka deyişle kişinin sunuya erişim yetkisini doğrulamasıdır. Şifre sorma işlemi kimlik doğrulamaya örnektir.

Erişim Denetimi (*Access Control*): Veriye olan erişimin denetlenebilir olmasıdır. Kullanıcı, kendisine verilen izinler doğrultusunda veriye ya da sunuya erişebilmekte ve işlem yapabilmektedir.

Veri Bütünlüğü (*Data Integrity*): Güvenlik terimleri kapsamında, veri bütünlüğü, gönderilen verinin yetkisiz kişiler tarafından değiştirilememesidir.

Şifreleme (*Encryption*): Bir iletinin içeriğini gizlemek amacıyla, iletinin kendisi üzerinde yapılan değişikliklere şifreleme denir. Bu tanımlamada sözedilen değişikler ile belirtilmek istenen, matematiksel bir algoritma ve şifreleme anahtarı kullanılarak bilginin alıcı dışındaki kişiler tarafından okunamaması ya da bilginin içeriğinin değiştirilememesi için kodlanmasıdır.

Güven Belgesi (*Credential*): Bir veriye ya da sunuya ulaşmak için istekte bulunan kişinin erişim izninin olduğunu gösteren belgedir. Güven belgelerine örnek olarak onay belgeleri, kullanıcı adı ve şifre verilebilir.

Rol (*Role*): Kullanıcılara verilmiş olan özel yetkiler grubudur.

İzin (*Permission*) / Hak (*Right*): Veriye ya da sunuya erişenin neleri yapabileceğini belirten durumdur.

Yasak (*Prohibition*) / Sakınım (Refrain) : Veriye ya da sunuya erişenin neleri yapamayacağını belirten durumdur.

Zorunluluk (*Obligation*): Veriye ya da sunuya erişenin neleri yapması gerektiğini belirten durumdur.

Özel İzin (*Dispensation*): Veriye ya da sunuya erişenin artık neleri yapmasına gerek kalmadığını belirten durumdur.

Yetki (*Authorization*): Veri ya da sunu üzerinde gerçekleştirilmesine izin verilen işlemlerdir.

Yetki Aktarımı (*Delegation*): Diğer varlık ya da varlıklar grubuna hak verilmesidir.

Konuşma Edimi (*Speech Act*): Politikaların daha az ayrıntılı olmasını ve merkezi olmayan güvenlik denetimini sağlar. Politikaların devingen olarak değiştirilmesine olanak verir. Konuşma edimlerinde

gönderici (*sender*), alıcı (*receiver*), içerik (*content*) ve koşul (*condition*) olmak üzere dört özellik bulunmaktadır. Konuşma edimlerinin geçerli olabilmesi için göndericinin uygun izinleri olmalıdır. Dört konuşma edimi vardır. Bunlar: Yetki Aktarımı, İstek, Feshetme ve İptal'dir.

- *Yetki Aktarımı (Delegate):* Gönderici alıcı için izin ekler.
- *İstek (Request):* Gönderici bir eylem ya da yetki için istekte bulunur.
- *Geri Alma (Revoke):* Gönderici bir izini siler ya da bir yasak ekler.
- *İptal (Cancel):* Gönderici isteği etkisiz kılar.

Politika Motoru (*Policy Engine*): Politika motoru politikaları yorumlar ve çıkarsama yapar. Konuşma edimleri ve etki alanı (*domain*) bilgisini kullanarak uygulanabilir. Yetkilerin, yasakların, zorunlulukların ve özel izinlerin kararlarını verir.

Çelişki Çözümü (*Conflict Resolution*): Aynı hedefteki aynı işlem için farklı kurallar/politikalar tanımlanmış olabilir. Böyle bir durumda çelişki meydana gelecektir. Birden fazla kuralın var olması durumunda hangi kuralın uygulanacağının ya da uygulanmayacağının belirlenmesi işlemi çelişki çözümüdür. Amaç çatışmanın sonlandırılmasıdır. Çözüm için öncelik belirlemesinden (*specifying priority*) ve öncelik ilişkilerinden (*precedence relations*) yararlanılmaktadır.

Üstpolitika (*Metapolicy*): Politikaların nasıl yorumlandığı (*interpreted*) ve çelişkilerin devingen olarak nasıl çözüldüğüne ilişkin politikalardır.

Politikaların Uygulanabilirliği (*Enforcement of Policies*): Çeşitli ortamlar için olan politika tanımlarının gerçekleştirilebilir diğer politikalara eşleştirilmesidir (*mapping*).

4.1.1 Politika örnekleri

Bu tez kapsamında tanımlanan 4 politika nesnesi olan izin, yasak, zorunluluk ve özel izin için sağlık etki alanı temel alınarak aşağıdaki politika örnekleri oluşturulmuştur.

- Eğer bir hasta şeker hastası ise "Diet.doc" dosyasını okuyabilir. (İzin)
- Bir doktor bir hastaya eğer o hastanın birincil doktoru ise reçete yazabilir. (İzin)
- Sadece hastanın kendi doktoru hastanın şeker hastası menüsünü değiştirebilir. (İzin)
- Bir doktor eğer hastanın resmi doktoru değil ise o hastaya reçete yazamaz. (Yasak)
- Bir şeker hastası günde 6 öğün yemelidir. (Zorunluluk)
- Bir şeker hastası alkollü içecekler içemez. (Yasak)
- Bir şeker hastası lifli yiyecekler yiyebilir. (İzin)

- Bir şeker hastası basketbol ve futbol gibi grup sporları yapamaz. (Yasak)
- Kan şekeri değeri normal olan bir şeker hastası ameliyat olabilir. (Özel İzin)
- Hastanın birincil doktoru kendi yokluğunda hastaya reçete yazabilmesi için ikincil bir doktora reçete yazma izini verebilir. (Konuşma edimi için izin)

İlerleyen alt bölümlerde Anlamsal Web politika dillerinin özellikleri ve kaynakçada yer alan politika dilleri anlatılmaktadır.

4.2 Anlamsal Web Politika Dilleri

Anlamsal Web'i güvenli hale getirmek için güvenlik gereksinimlerini karşılayan bir anlamsal politika diline gereksinim vardır. Politikalar Anlamsal Web uygulamaları için önemlidir. Farklı şekillerde belirtilebilecekleri çeşitli yaklaşımlar bulunmaktadır. Ancak, herhangi bir politika gösteriminde karşılanması gereken bazı genel gereksinimler bulunmaktadır: yönetilen sistemde ortaya çıkan çeşitli politika gereksinimlerini işlemek için *anlamlılık (expressiveness)*, farklı derecelerde uzmanlığı olan yöneticilerin politika tanımı görevlerini kolaylaştırmak için *kolaylık (simplicity)*, farklı düzlemler için politika belirtimlerinin uygulanabilir politikalara eşlemesini sağlamak için *zorlanabilirlik (enforceability)*, uygun başarımı sağlayabilmek için *ölçeklenebilirlik (scalability)* ve politikalar üzerine akıl yürütmek için *çözümlenebilirlik (analyzability)* (Tonti et al., 2003). Bir politika dili; iyi

tanımlanmış (*well-defined*), esnek (*flexible*), genişletilebilir (*extensible*) ve diğer diller ile birlikte çalışabilir (*interoperable*) olmalıdır. Bir politika dili olayları daha iyi bir şekilde belirtmek için açıklayıcı dilin (*declarative language*) gücüne gereksinim duyar.

Anlamsal Web teknolojilerine dayanan politika dilleri, politikaların çok çeşitli etki alanı verileri üzerinde tanımlanmasına izin vermekte ve aynı bilgi modelini kullanmayan katılımcılar arasında ortak anlamı desteklemektedir (Finin et al., 2008). Son yıllarda yapılan erişim denetimi çalışmalarında iki paralel konu ele alınmaktadır (Finin et al., 2008): gerçek dünya uygulama etki alanlarının politika gereksinimlerini karşılamaya yönelik erişim denetim modellerinin geliştirilmesi ve erişim denetimi için politika dillerinin geliştirilmesi. Bu iki paralel konunun, erişim denetimi ve politika dilleri, güvenlik altyapısının gelişimini sağlamak için görevdeşlik yaratması gerektiği düşünülmektedir. Bu tez çalışması, gerçek dünya uygulama etki alanlarının politika gereksinimlerini karşılamaya yönelik ontoloji tabanlı bir erişim denetim modelinin geliştirilmesi üzerine odaklanmaktadır.

Anlamsal Web politika dillerinin bazıları şunlardır: Rei[1], KAoS[2], Ponder[3], Rein[4], XACML[5], Proteus (Toninelli et al., 2007), WSPL (Anderson, 2004), Protune[6] ve Appel[7]. Bu tez çalışmasında kullanılacak olan politika dili Rei'dir.

4.2.1 Rei

Rei, OWL-Lite temelli bir politika tanımlama dilidir. Kullanıcıların yetkiler, yasaklar, zorunluluklar ve özel izinler kavramlarını tanımlamasına izin vermektedir (Tonti et al., 2003; Kagal et al., 2003). Rei, politika geliştiricilerinin, politikaları etki alanına özgü ontolojiler üzerinde RDF ve OWL gibi diller kullanarak tanımlamasına izin vermektedir. Sistemdeki yetkiler ve zorunlulukların varlıklar arasında değiş tokuş edilebilmesi için Rei politika dilinin konuşma edimleri kümesi vardır. Rei, politika tanımlamalarını çıkarsamak için de bir

---

[1] Rei, http://rei.umbc.edu/ .

[2] KaoS, http://www.ihmc.us/research/projects/KAoS/ .

[3] Ponder, http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml .

[4] Rein, http://dig.csail.mit.edu/2006/06/rein/ .

[5] XACML, http://www.oasis-open.org/committees/xacml/ .

[6] Protune, http://rewerse.net/I2/software.html .

[7] Appel, http://www.w3.org/TR/P3P-preferences/ .

Prolog politika motorunu kullanmaktadır. Rei politika motorunun saptadığı politika çelişkilerini çözmek için üstveri kullanılmaktadır. Politika motoru politika tanımlarını Rei ontolojisi ile tutarlı olacak şekilde hem Rei dilinde hem de RDF-S olarak alabilmektedir (Tonti et al., 2003). Şekil 4.1 Rei politika yapısını göstermektedir.

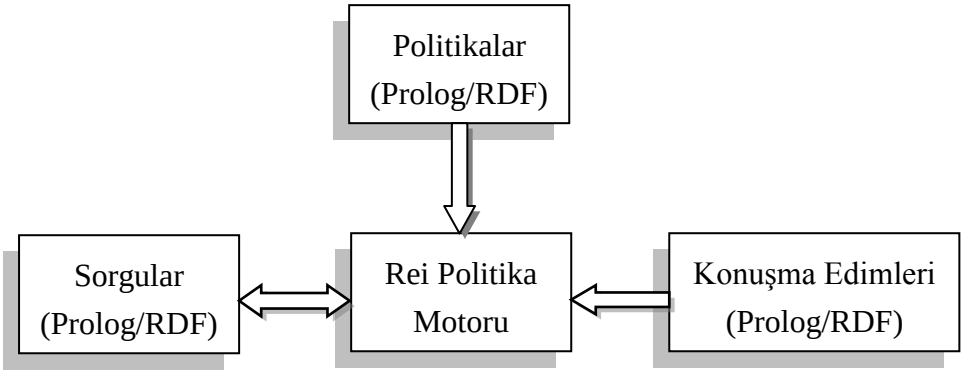
Şekil 4.1 Rei politika yapısı.

Rei motoru çok çeşitli sorgulara yanıt verebilir (Kagal et al., 2004):

- X'in Z kaynağı üzerinde Y eylemini gerçekleştirme izni var mı?
- X'in varolan zorunlulukları nelerdir?
- X, Z kaynağı üzerinde hangi eylemleri gerçekleştirebilir?
- Varolan politika etki alanında X'in bütün izinleri nelerdir?
- X'in, hangi koşullar altında Z kaynağında Y eylemini gerçekleştirme izni vardır?

Rei motoru bu sorgulara yanıt verirken konuşma edimlerini göz önüne alır ve üst politikaları kullanarak ortaya çıkabilecek çelişkileri çözmeye çalışır. Sorguların çalıştırılması, çelişkilerin çözümü ve

konuşma edimleri sırası ile 8.3.5, 8.3.6 ve 8.3.7 bölümlerinde anlatılmaktadır. Şekil 4.2'de Rei politika motoru yapısı yer almaktadır (Kagal, 2002).



Şekil 4.2 Rei politika motoru yapısı.

Aşağıda eğer kullanıcı e-ödeme sunusu için izin verilen grupta ise bu sunuya erişime izin veren bir Rei politika tanıtıcı örneği yer almaktadır (Clemente et al., 2005):

```
<constraint:SimpleConstraint rdf:ID="IsPayCustomer"
  constraint:subject="#RequesterVar"
  constraint:predicate="&example;memberOf"
  constraint:object="&example;payCustomer"/>
<constraint:SimpleConstraint rdf:ID="IsPayServer"
  constraint:subject="#PayServerVar"
  constraint:predicate="&example;memberOf"
  constraint:object="&example;payServer"/>
<constraint:And rdf:ID="ArePayCustomerAndPayServer">
  constraint:first="#IsPayCustomer"
  constraint:second="#IsPayServer"/>
<deontic:Permission rdf:ID="PayServerPermission">
  <deontic:actor rdf:resource="#RequesterVar"/>
  <deontic:action rdf:resource="&example;access"/>
  <deontic:constraint
      rdf:resource="#ArePayCustomerAndPayServer"/>
</deontic:Permission>
<policy:Policy rdf:ID="PaymentAuthPolicy1">
```

```
  <policy:grants rdf:resource="#PayServerPermission"/>
</policy:Policy>
```

4.2.2.1 REI terimbilimi

Rei politika dilinde kullanılan terimbilimi Çizelge 4.1'de gösterilmektedir.

Çizelge 4.1 Rei terimleri.

Rei Terimi	***Tanım***
politika ***policy***	Etki alanı içerisindeki varlıkların davranışlarını tanımlayan kurallar kümesidir.
politika etki alanı ***policy domain***	Politika tarafından etkilenen varlıklar ve kaynaklar kümesidir.
deontik nesne ***deontic object***	Deontik nesneler: İzin (*permission*), yasak (*prohibition*), zorunluluk (*obligation*) ve özel izindir (*dispensation*).
konuşma edimi ***speech act***	Bazı eylemleri yerine getirmek için dilin kullanımıdır.
üst politika ***meta policy***	Bir politikanın nasıl uygulanacağını belirten bir politikadır. Rei politika dilinde çelişkilerin çözümlenmesi için kullanılan düzenekleri belirtir.
kısıt ***constraint***	Belirli bir politika, kural ya da deontik nesne ile ilgili bir koşuldur.

4.2.2.2 REI ontolojileri

Her bir Rei ontolojisi etki alanı ile ilgili olan sınıfları (*class*) ve özellikleri (*property*) tanımlamaktadır. Rei politika dili aşağıdaki

ontolojilerden oluşmaktadır:

- ReiPolicy
    - Politika etki alanı içerisindeki varlıkların davranışlarını belirler. Kuralları ve politika etki alanını tanımlayan bağlamı içerir.
- ReiMetaPolicy
    - Üst politikalar, politikaların nasıl yorumlandığı (*interpreted*) ve çelişkilerin nasıl çözümlendiği ile ilgili politikalardır. Rei, iki üst politika türünü modellemektedir: (i) varsayılanlar (ii) politikaların farklı gereksinimlerini karşılamaya yönelik olan çelişkilerin çözümlenmesi. Varsayılanlara yönelik olan üst politikalar `Behavior` ve `MetaMetaPolicy` sınıflarını, çelişkilerin çözümüne yönelik olan üst politikalarda `Priority` ve `ModalityPrecedence` sınıflarını içermektedir.
- ReiEntity
    - Herhangi bir kullanıcı, yazılım etmeni veya donanım kaynağı `entity:Entity` ile tanımlanmaktadır.

- ReiDeontic
    - Politika etki alanındaki varlıklar üzerinde izinler, yasaklar, zorunluluklar ve özel izinler yaratılması için kullanılmaktadır. Burada; hangi aktör ya da aktörler kümesinin hangi eylem ya da eylemler

kümesini hangi kısıtlar altında gerçekleştireceği tanımlanmaktadır.

- ReiConstraint
  - Koşul, nesneler kümesini ve eylemler kümesini tanımlamak için kullanılmaktadır. İki çeşit koşul tanımlanmaktadır: `SimpleConstraint` ve `BooleanConstraint`. `SimpleConstraint`, RDF deyimleri gibi üçlü olarak biçimlenmektedir: özne (*subject*), yüklem (*predicate*) ve nesne (*object*). `BooleanConstraint`'de ise hem Simple hem Boolean koşulları `And`, `Or` ve `Not` boole işleci kullanılarak birleştirilebilir.
- ReiAnalysis
  - Tutarlı ve geçerli politikalar geliştirebilmek için kullanılan bu ontolojide Rei iki belirtim sağlamaktadır: `UseCase` yönetimi ve `WhatIf` çözümlemesi. `UseCase` yönetiminde, diğer politikalar kümesine karşı bir grup `UseCase`'in doğruluğu daima sağlanmaktadır. `WhatIf` çözümlemesi, politikada ya da varlıklarda geçici olarak meydana gelen değişiklikleri işleme koymadan önce etkilerini sınamak amacı ile kullanılmaktadır.
- ReiAction
  - Politikalar etki alanındaki eylemler üzerinden

tanımlanmaktadır. Bu ontoloji, bütün eylemler için gerekli olan özellikleri içermekte, ayrıca eylemler ile ilgili olan etki alanına bağlı bilgilerin eklenmesine de izin vermektedir.

Ek-1'de Rei ontolojileri ve bu ontolojilerin sınıf ve özellik sıradüzenselleri görülmektedir.

4.2.2 KAoS

KAoS politika dilinde ontolojiler OWL dili ile tanımlanmaktadır. Örgü sunuları için politika ve etki alanı yönetimi sunularından oluşmaktadır. KAoS Politika Ontolojisi (KPO), yetkiler (eyleme izin veren ya da yasaklayan kısıtlar) ve zorunluluklardan (bir durum meydana geldiğinde bazı eylemleri gerektiren ya da bu gereksinimden vazgeçilmesini belirten kısıtlar) oluşmaktadır (Tonti et al., 2003). KAoS önce KPO'yu daha sonra ek ontolojileri yükler. Türdeş politika temsili, kapsamlılığı, ölçeklenebilirlik ve başarım KaoS'un önemli özellikleridir (Uszok et al., 2004).

KAoS, Sun'ın Java Etmen Sunularını (Java Agent Services - JAS) temel almaktadır ve çıkarsama için JTP'yi (Java Theorem Prover) kullanmaktadır (Uszok et al., 2004).

KaoS grafiksel arayüz olarak KaoS Politika Yönetim Aracını (KaoS Policy Administration Tool - KPAT) sağlamaktadır. KPAT

kullanıcılara politika tanımlamasında, düzeltme ve uygulamada yardımcı olmaktadır. Ayrıca, ontolojilere göz atmak ve ontolojilerin yüklenmesinde ve yeni tanımlanmış ontolojilerin çözümlemesinde ve çelişkilerin çözümünde de kullanılmaktadır (Tonti et al., 2003).

KAoS ve Rei politika dilleri arasındaki temel fark, KAoS politika dili betimleme mantığını temel alırken Rei politika dilinin bilişimsel mantığı temel almasıdır.

Aşağıda eğer kullanıcı e-ödeme sunusu için izin verilen grupta ise bu sunuya erişime izin veren bir KAoS politikası örneği yer almaktadır (Clemente et al., 2005):

```
<owl:Class rdf:ID=”PaymentAuthAction”
  <owl:intersectionOf rdf:parse Type=”owl:collection”>
     <owl:Class rdf:about=”&action;AccessAction”/>
       <owl:Restriction>
         <owl:onProperty
           rdf:resource=”&action;#performedBy”/>
         <owl:toClass
           rdf:resource=”&domains;MembersOfPayCustomer”/>
       </owl:Restriction>
       <owl:Restriction>
         <owl:onProperty
           rdfresource=”&action;#performedOn”/>
         <owl:toClass
           rdf:resource=”&domains;MembersOfPayServer”/>
     </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
<policy:PosAuthorizationPolicy rdf:ID=”PaymentAuthPolicy1”>
  <policy:controls rdf:ID=”PaymentAuthAction”/>
  <policy:hasSiteOfEnforcement rdf:resource=”#TargetSite”/>
  <policy:hasPriority>1</policy:hasPriority>
</policy:PosAuthorizationPolicy>
```

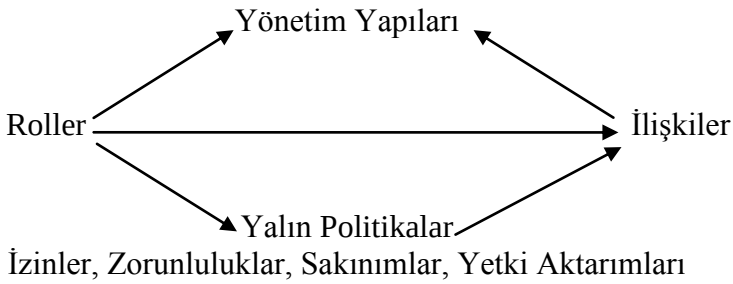
4.2.3 Ponder

Ponder bildirim deyimlerinden oluşan nesneye dayalı bir politika dilidir. Ponder politika yönetimi için yöntemleri desteklemektedir. Politikaları hazırlamak, güncelleştirmek, silmek ve taramak için çeşitli grafiksel araçlar sağlamaktadır. Politika tanımlamalarının sözdizimsel ve anlamsal çözümlemesi için, çalışma zamanında yorumlanabilecek Ponder dili tanımlamalarını XML ya da Java koduna dönüştürmek için araçlar bulunmaktadır (Tonti et al., 2003).

Ponder'da, Rei politika dilinde yer almayan, yalın ve bileşik politikalar bulunmaktadır. Yalın politika (*basic policy*) sistem davranışlarında yer alan seçimleri yöneten bir kural olarak düşünülmektedir ve özneler kümesi ile hedefler kümesi arasında bir bildirim ile belirtilmektedir. Bu kümeler politikanın üzerinde çalıştığı yönetilen nesneleri tanımlamak için kullanılmaktadır (Tonti et al., 2003). Bileşik (*composite*) politikalar yalın politikaların gruplanmasından oluşmaktadır.

Ponder'da dört politika türü bulunmaktadır. Bunlar; izinler (*authorizations*), zorunluluklar (*obligations*), sakınımlar (*refrains*) ve yetki aktarımları (*delegations*)'dır. İzinler; bir öznenin hedef nesne üzerinde gerçekleştirmesine izin verilen işlemlerdir (Quinn et al., 2004). Zorunluluklar; belirli bir olay meydana geldiğinde öznenin hedef nesne üzerinde yerine getirmesi gereken eylemlerdir. Sakınımlar; bir öznenin yapmasına izin verilmeyen eylemleri tanımlamaktadır. Sakınımlar bir

politikanın yürütülmesinde hedefe güvenilmediğinde kullanılmaktadır. Diğer varlığa ya da varlıklar grubuna yetki verilmesi ise yetki aktarımlarıdır. Roller, ilişkiler ve yönetim yapıları bileşik politikaları oluşturmak için kullanılan üç bileşik politika türüdür (Dulay et al., 2001). Şekil 4.3'de Ponder politika yapısı yer almaktadır.



Şekil 4.3 Ponder politika yapısı.

İzin politikası söz dizimi aşağıdaki gibidir:

```
inst ( auth+ | auth- ) policyName "{"
     subject    [<type>]   domain-Scope-Expression;
     target     [<type>]   domain-Scope-Expression;
     action     action-list;
     [ when     constraint-Expression; ]   "}"
```

Aşağıdaki örnekler olumlu ve olumsuz izin politikalarını göstermektedir (Damianou et al., 1995).

```
inst auth+ switchPolicyOps {
     subject    /NetworkAdmin;
     target     [PolicyT]  /Nregion/switches;
     action     load(),remove(),enable(),disable();
}
```

// NetworkAdmin etki alanı üyelerinin Nregion/switsches etki

alanında PolicyT türündeki nesneleri yükleme (load), silme (remove), seçme (enable) ve seçilemez (disable) eylemlerini gerçekleştirme izni vardır.

```
inst auth- /negativeAuth/tstRouters {
      subject    /testEngineers/trainee;
      target     [routerT]  /routers;
      action     performance_test();
}
```

// Stajyer sınama mühendislerinin yöneticiler üzerinde başarım sınamaları yapmaları yasaklanmıştır.

4.2.4 Rein

Rein (Rei ve N3), Anlamsal Web'de dağıtık politikaların tanıtımında ve çıkarsamasında kullanılan dağıtık bir politika çatısıdır. Rein, politikalar için Rei kavramlarını ve politikaları birbirleri ve örgüdeki politika ağları ile bağlamak için Anlamsal Web kural dili olan N3[1] (Notation 3) kurallarını kullanmaktadır. Politika kararlarını elde etmek için yapılan çıkarsama işlemi bir N3 çıkarsama aracı olan cwm[2] (*closed worl machine*) kullanılarak gerçekleştirilmektedir. Rein üç temel bileşenden oluşmaktadır; politika ağlarını tanımlamak için üst düzey bir *ontoloji*, bilgiyi kullanmak ve politikalar ile örgü kaynaklarından çıkarsama yapmak için *N3 düzenekleri*, politika ağlarından politika

[1] N3, http://www.w3.org/DesignIssues/Notation3.html.

[2] CWM, http://www.w3.org/2000/10/swap/doc/cwm.html.

kararlarını çıkaran bir *motor* (Kagal and Finin, 2005).

Rein, politikaların RDF-S ve OWL kullanan ontolojiler ile tanımlanmasına izin verir. Ayrıca, politikaların N3 kurallarında belirtilen düzenekleri kullanarak varlıklar ile diğer politikalarda ve kaynaklarda tanımlanmış olan ilişkilerle ilgili veri ve yorumları kullanmasına izin verir. Bu politikalar ve kaynaklar erişim bağları ile örtülü olarak bağlanmıştır ve Rein politika ağlarını oluşturmaktadır. Böylece bir politikanın ontolojiyi veya başka bir politika tarafından kullanılan akıl yürütmeyi ya da kendi kuralları içinden kullanabilmek için bir örgü kaynağını anlamasına gereksinim duymadığı birimsel bir yapı elde edilmektedir (Kagal and Finin, 2005). Politika ağları, Anlamsal Web'de dağıtık olarak bulunan diğer politikalardan ve bilgiden politikaların tanımlanmasına izin vermektedir (Kagal and Finin, 2005).

Rein, politika tanımlamasında tek bir politika ontolojisi ya da dili ileri sürmez. Her politikanın kendi politika ontolojisine ve eğer gereksinim olursa politika ontolojisinin anlamsallığını yorumlamak için N3 kuralları için bir çıkarsayıcısının olmasına izin verir. Politikalar yalından karmaşığa doğru sınıflandırılabilir (Kagal and Finin, 2005).

Aşağıda kullanıcının group.jpg dosyasına erişim isteğini belirten örnel bir Rein politikası tanımlanmıştır[1]:

```
@prefix : <http://dig.csail.mit.edu/2005/09/rein/network#> .
@prefix http: <http://dig.csail.mit.edu/2005/09/rein/
```

[1] Rein politika örneği, http://dig.csail.mit.edu/2006/06/rein/#example .

```
            examples/http-access#> .
[     a :Request;
        :access http:can-get;
        :ans :Valid;
        :requester  [
          http:can-get <http://demo.policyawareweb.org/
                              images/group.jpg> ];
        :resource <http://demo.policyawareweb.org/
                    images/group.jpg> ].
```

4.2.5 XACML

XACML (Extensible Access Control Mark-up Language), erişim denetim politikalarını göstermek için kullanılan XML tabanlı bir dildir. XACML[1], XML olarak tanımlanmış nesnelere karşı, yetki politikalarının XML olarak tanımlanması için tasarlanmıştır. Bir XACML politikası temel bileşenleri hedef, etki ve koşullar olan kurallar kümesinden oluşur. Hedef, kuralın uygulanacağı kaynaklar, özneler ve eylemler kümesini tanımlamaktadır. Kuralın etkisi izin ya da yok saymak olarak olacaktır. Koşul, kuralın uygulanabilirliğini belirten bir boole tanımı gösterir. Bir istek, istek ile ilgili öznenin, istekte yer alan kaynağın, yerine getirilen eylemin ve çevrenin ilişkili olduğu öznitelikleri içerir. Yanıt ise dört karardan birini içerir: izin (*permit*), yok saymak (*deny*), uygulanamaz (*not applicable*), belirsiz (*indeterminate*). Uygulanamaz kararı, uygulanabilecek politikaların ya da kuralların bulunamadığı durumu; belirsiz kararı ise erişim denetim işlemi sırasında bazı hataların meydana geldiğini belirtir. Bir istek, bir politika ve ilgili yanıt XACML bağlamını (*XACML context*) oluşturur (Damiani et al., 2004).

[1] http://www.oasis-open.org/committees/xacml

Kuralların uygulandığı özneler ve kaynaklar önceden tanımlanmış işlevler (örneğin; eşitlik, küme karşılaştırma, aritmetik) ve veri türleri (örneğin; tamsayı, boole, karakter dizisi) ile tanımlanmaktadırlar. Aşağıdaki XACML örnek kural parçası Trauma Terapi Merkezi'nin doktorlarının hastalara yardım eden doktorların videosunu izleyebilir iznini göstermektedir (Damiani et al., 2004):

```
<?xml version="1.0" encoding="UTF-8"?>
<Rule
      xmlns="urn:oasis:names:tc:xacml:1.0:policy"
      xmlns:xsi= http://www.w3.org/2001/XMLSchema-instance
      xmlns:ctx="urn:oasis:names:tc:xacml:1.0:context"
      xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
      xmlns:md="http://ourdomain.it/MD/Schema/md-syntax"
      xmlns:ms="http://ourdomain.it/MS/Schema/ms-syntax"
      RuleId="urn:oasis:names:tc:xacml:examples:ruleid:1"
      Effect="Permit">
      <Target>
        <Subjects>
          <Subject>
           <SubjectMatch
               MatchId= "urn:ourdomain:function:metadataQuery">
               <AttributeValue
                DataType="http://">
                <rdf:Statement rdf:about="thisRequestUser" >
                  <rdf:subject rdf:resource="http://ourdomain.it/
                                MS/Schema/ms-syntax#Physician" />
                  <rdf:predicate rdf:resource="http://ourdomain.
                                it/MS/Schema/ms-syntax#belongs"/>
                  <rdf:object rdf:datatype="http://www.w3.org/
                                2001/XMLSchema#string">
                     Trauma Therapy Center
                  </rdf:object>
                </rdf:Statement>
               </AttributeValue>
              <SubjectAttributeDesignator
                AttributeId="urn:ourdomain:attribute:metatag"
                DataType="http://www.w3.org/2001/XMLSchema#string
                        "/>
           </SubjectMatch>
          </Subject>
         </Subjects>
         <Resources>
          <Resource>
           <ResourceMatch
```

```
      MatchId="urn:ourdomain:function:metadataQuery">
      <AttributeValue
      DataType="http://">
      <rdf:Statement rdf:about="thisRequestUrl">
        <rdf:subject rdf:resource="http://ourdomain.it/
                    MS/Schema/md-syntax#Video"/>
        <rdf:predicate rdf:resource="http://ourdomain.it/
                      MD/Schema/md-syntax#shows how"/>
        <rdf:object rdf:nodeID="content"/>
      </rdf:Statement>
      <rdf:Statement rdf:nodeID="content">
        <rdf:subject rdf:resource="http://ourdomain.it/
                    MS/Schema/ms-syntax#Physician"/>
        <rdf:predicate rdf:resource="http://ourdomain.it/
                      MS/Schema/ms-syntax#assists"/>
        <rdf:object rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#string">
                Patient
        </rdf:object>
      </rdf:Statement>
      </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:ourdomain:attribute:metatag"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ResourceMatch>
    </Resource>
   </Resources>
   <Actions>
    <Action>
     <AnyAction />
    </Action
   </Actions>
  </Target>
 </Rule>
```

Bu kural, öznesinin öznitelik `rolü (role)` olarak değeri `physician` olan istekleri eşlemektedir. Bu işlevler ve veri türleri birçok erişim denetim politikasının tanımlanması için kullanılabilirler. Ek olarak işlevler ve veri türleri tanımlanması için bir XACML ayrıca genişleyebilen bir düzenek belirtmektedir (Damiani et al., 2004).

4.2.6 Proteus ve Anlamsal Bağlam Farkındalığı Yaklaşımı

Bağlam (*context*), bir varlığın durumunu ya da çalışmasını veya bu varlığın işlem yaptığı dünyayı niteleyen herhangi bir bilgi olarak tanımlanmaktadır (Toninelli et al., 2007). Proteus bağlam ve politika modeli, varlık ve eylem kavramlarını kullanan bir dizgede meydana gelen etkileşimleri tanımlayan bir dizge modelini temel almaktadır. Varlık, dizgedeki herhangi bir aktörü veya kaynağı belirtir ve öznitelik-değer çiftleri kullanılarak tanımlanan bir dizi özellik ile açıklanmaktadır. Eylem, bir varlık ya da aktör tarafından gerçekleştirilebilecek bir çalışmayı tanımlar. Eylem, eylem bağlamı olarak tanımlanan belirli bir işletim durumu içinde gerçekleştirilir. Eylem bağlamı, eylemi niteleyen öznitelikleri içerir. Örneğin, eylemin gerçekleştirileceği hedef ve eylemi gerçekleştiren varlık. Bir varlık ve bir eylem arasındaki ilişki ise etkileşim olarak tanımlanmaktadır (Toninelli et al., 2007).

Proteus çatısı, üç farklı uyarlamayı sağlamak için bağlam farkındalığı (*context-aware*) ve anlamsal teknolojileri kullanmaktadır: politika uyarlaması, eylem uyarlaması ve bağlam uyarlaması. Uyarlama terimi, farklı ve olası beklenmedik durumlarda politikaların yürütülmesini sağlayabilmek için bağlam ve politika tanımlarını ayarlayan politika tabanlı yönetim dizgelerinin yeteneğini belirtir. Politika uyarlaması, etkin bağlamında değişiklik olan etkin bir politikanın geçerliliğini kendiliğinden uzatmaktadır. Eylem uyarlaması, varolan durum için gerçekleştirilemeyen izin/zorunluluk eylemleri yerine diğer izin/zorunluluk eylemlerinin bulunmasıdır. Bağlam uyarlaması,

izin/zorunluluk eylemlerinin gerçekleştirilebileceği diğer bağlam tanımlanmasını göstermektedir. Bağlam uyarlaması devingen politika çelişkilerinde yardımcı bir durumdur. Örneğin, bir varlığın kendisine yasak olan bir eylemi gerçekleştirme zorunluluğu var ise izin/zorunluluk eylemlerini değiştirmek yerine, Proteus, eylemlerin gerçekleştirilebileceği farklı bir bağlam tanımlamaktadır (Toninelli et al., 2007).

Proteus bağlamı, politikalar ile ilgili olan bütün bilgileri içerir. Bunlar; kaynağın yükü ya da kullanılabilirliği gibi kaynakların durumu, kaynaklarda işlem gerçekleştirmek isteyen aktörler ve bu aktorlerin rolleri ya da kimlik bilgileri, zaman ve diğer kullanılabilir kaynaklar gibi bilgileri içeren çevre koşullarıdır. Proteus, bağlamı bir öznitelikler kümesi ve önceden belirlenmiş değerler şeklinde modeller. Tek bir değerin tersine bir öznitelik ayrıca izin verilen değerler dizisi için kısıtlar tanımlayabilir. Bir öznitelik değeri, bir değişmez veya bir değişken olabilir (Toninelli et al., 2007).

Proteus modelinde, ontolojiler OWL kullanılarak geliştirilmiştir. Aşağıda iki Proteus politika örneği yer almaktadır (Toninelli et al., 2007).

Toplantı kaynaklarına erişimi denetleyen bir erişim denetim politikası:

$$MeetingPolicy = Pos\_Authorization\_Policy \sqcap \exists controls.MeetingAction \sqcap \exists activating\_context.Meeting\_Context$$

Kullanıcı işvereninden gelen bir çağrıya yanıt veremediğinde işverenine göndermek zorunda olduğu ileti ile ilgili bir Proteus zorunluluk politikası:

$$
\begin{aligned}
&Boss\_Notification\_Policy \\
&\quad = Obligation\_Policy \\
&\quad \sqcap \exists triggers.SendSMS\_to\_Boss\_Action \sqcap \\
&\quad\ \ \exists activating\_context.No\_Answer\_Boss\_Context
\end{aligned}
$$

4.2.7 Protune

Protune (PROvisional TrUst Negotiation) REWERSE[1] projesinin politika ve üst dilidir. Bu politika dili erişim denetimi politikaları, gizlilik politikaları, geçici politikalar ve iş kurallarının belirtilmesinde kullanılmaktadır. Protune, varolan durumu değiştiren eylemleri tanımlayan bildirim deyimi dilidir (Bonatti and Olmedilla, 2005). Protune'da yer alan güven uzlaşmasında PAPL (Bonatti and Samarati, 2000) ve PeerTrust'dan esinlenilmiştir (Bonatti and Olmedilla, 2005).

Protune politika dili nesneye dayalı sözdizimi ile geliştirilmiş bir mantıksal programlama dilidir. Örneğin; kredi kartı kullanarak bir kitap alınmasına izin veren kural aşağıdaki kurallar kümesi ile tanımlanmaktadır (De Coi et al., 2008):

[1] REWERSE, http://cs.na.infn.it/rewerse/.

$allow(buy(Resource))$
$\leftarrow credential(C), valid\_credit\_card(C), accepted\_credit\_card(C).$

$valid\_credit\_card(C) \leftarrow C.expiration: Exp, Date(Today), Exp > Today.$

Burada *C.expiration : Exp* , *Exp*'in *C*'nin *expiration* özelliğinin değerini belirten nesneye dayalı bir gösterimdir.

4.2.8 Appel

Appel (A P3P Preference Exchange Language), P3P (Platform for Privacy Preferences) etmenleri arasında P3P politikalarını göz önüne alarak seçenekler derlemini tanımlayan bir dildir. P3P, kullanıcıları sunuların (gizlilik bildiren örgü sayfaları ve uygulamaları) gizlilik politikalarına ilişkin bilgilendirmektedir. Bu dil kullanılarak kullanıcı kendi seçeneklerini seçenek-kural (kural kümesi denen) kümesinde göstermektedir. Ayrıca, daha sonra kullanıcı etmeni tarafından, P3P kullanan örgü sayfalarının makine-okunabilir gizlilik politikalarının benimsenebirliği ile ilgili olarak özdevimleştirilmiş ya da yarı-özdevimleştirilmiş kararları vermek içinde bu dil kullanılmaktadır (Cranor et al., 2002).

Appel, kullanıcıların diğer gruplar tarafından yaratılmış olan seçenek kural kümelerini aktarmalarına izin vermek ve bir çok kullanıcı etmeni arasında kendi kural kümesi dosyalarını taşımak için de kullanılmaktadır. Ayrıca, kullanıcı etmeni gerçekleştiriciler, kullanıcı

etmenlerinin karar verme bileşeni olarak çalışan kural değerlendiriciler tarafından kullanıcı seçeneklerini kodlamak için bu dili kullanmayı seçebilirler (Cranor et al., 2002).

APPEL kural kümeleri P3P politikaları üzerinden seçeneklerin tanımlanması için tasarlanmış olmasına rağmen APPEL'in sözdiziminin ve anlamsallığının büyük bir kısmı P3P 1.0 Belirtiminden etkilenmektedir (Cranor et al., 2002). Aşağıda APPEL 1.0 için yalın bir kural kümesi örneği verilmiştir. Bu örnekte kullanıcı doğum günü bilgisini isteyen sunuların bu bilgiyi yasal süreden daha uzun süre tutamayacağını belirtmektedir (Duma et al., 2007).

```
<appel:RULE behavior="request"
     description="Must not keep">
     <p3p:POLICY>
          <p3p:STATEMENT>
               <p3p:DATA-GROUP>
                    </p3p:DATA ref="#user.bdate"/>
               </p3p:DATA-GROUP>
               <p3p:RETENTION>
                    <legal-requirement/>
               </p3p:RETENTION>
          </p3p:STATEMENT>
     </p3p:POLICY>
</appel:RULE>
```

4.2.9 WSPL

Web Sunuları Politika Dili (Web Services Policy Language - WSPL) yetki, hizmet niteliği (QoS), gizlilik ve uygulamaya özel sunu seçenekleri gibi çok çeşitli politikalar belirtmek için uygun bir dildir. WSPL sözdizimi OASIS XACML (eXtensible Access Control Markup

Language) standardının bir alt kümesidir (Godik and Moses, 2003). WSPL, uygulanmakta olan ve örgü sunuları ile birlikte kullanılan standart bir politika dilidir.

WSPL politikası bir ya da daha fazla kural dizisinden oluşmaktadır. Her bir kural politikayı sağlayabilmek için benimsenebilir bir seçimi göstermektedir. Kurallar en çok seçilen seçim ilk listelenmek üzere seçenekler sırasına göre listelenmektedir (Anderson, 2004). Aşağıda Kerberos'un ya da X509'un kullanılması gerektiğini berliten bir WSPL politika örneği verilmiştir[1]:

```
<Policy PolicyId="policy:1" RuleCombiningAlgorithm="&permit-
     overrides;">
     <Rule RuleId="rule:1" Effect="Permit">
       <Condition FunctionId="&function;string-is-in">
          <AttributeValue
          DataType="&string;">Kerberosv5TGT</AttributeValue>
          <ResourceAttributeDesignator
               AttributeId="&SecurityToken;"
               DataType="&string;"/>
       </Condition>
     </Rule>
     <Rule RuleId="rule:2" Effect="Permit">
       <Condition FunctionId="&function;string-is-in">
          <AttributeValue
          DataType="&string;">X509v3</AttributeValue>
          <ResourceAttributeDesignator
               AttributeId="&SecurityToken;"
               DataType="&string;"/>
       </Condition>
     </Rule>
</Policy>
```

[1] WSPL politika örneği, http://research.sun.com/projects/xacml/WSPL_vs_WS-Policy_v2.pdf .

4.3 Anlamsal Web Politika Dillerinin Karşılaştırılması

Bu bölümde KAoS, Rei, Ponder, Protune, XACML ve WSPL dilleri karşılaştırılmaktadır. Çizelge 4.2'de bu politika dillerinin karşılaştırması yer almaktadır (De Coi and Olmedilla, 2008). Çizelgede karşılaştırma; iyi tanımlanmış alamsallık, altyapıdaki biçim, eylemin yürütülmesi, yetki aktarımı, gerçekleştirim türü, uzlaşma, sunuş biçimi ve genişletilebilirlik açılarından yapılmaktadır.

İyi Tanımlanmış Anlamsallık: Bir politika dilinin anlamsallığı eğer o dilde oluşturulmuş olan politikanın anlamı dilin gerçekleştiriminden bağımsız ise iyi tanımlanmıştır. (De Coi and Olmedilla, 2008) çalışmasında eğer bir politika dili Mantık Programlama veya Betimleme Mantığı (Description Logic - DL) temelli ise iyi tanımlanmış denilebilir. Bu durumda KAoS, Rei ve Protune politika dilleri iyi tanımlanmış anlamsallığı taşır. Bu nedenle, Rei politika dilinin kullanıldığı Ontoloji Tabanlı Erişim Denetimi ile iyi tanımlanmış anlamsallığı taşıyan politikalar oluşturulmaktadır.

Altyapıdaki Biçim: Anlamsal olarak iyi tanımlanmış dillerden KAoS Description Logic temelli; Rei, Mantık Programlama, Betimleme Mantığı ve Deontik Mantığı birleştirmekte; Protune Mantık Programlama temellidir. Ponder ise nesneye dayalı bir politika dilidir. WSPL ve XACML için bir biçim yoktur.

Eylemin Yürütülmesi: Ponder, saat bilgisi gibi dizge özelliklerine politika içerisinden ulaşılmasına izin vermektedir. Ayrıca, herhangi bir zorunluluk meydana geldiğinde hangi eylemin gerçekleştirileceğini belirtmektedir. XACML, politika içerisinden eylemlerin tanımlanmasına izin vermektedir. Bu eylemler politikanın değerlendirilmesi sırasında toplanmakta ve istekte bulunana bir yanıt gönderilmeden önce gerçekleştirilmektedir (De Coi and Olmedilla, 2008). Aynı düzenek WSPL tarafından da sağlanmaktadır. Protune politika dili eylemin yürütülmesini desteklerken KAoS ve Rei politika dilleri desteklememektedir.

Yetki Aktarımı: Ponder, yetki aktarımı için özel bir politika türü tanımlamaktadır. Rei ve Protune politika dilleri de yetki aktarımını desteklemektedir. KAoS, XACML ve WSPL ise yetki aktarımını desteklememektedir. Ontoloji Tabanlı Erişim Denetimi yetki aktarımını sağlamaktadır.

Gerçekleştirim Türü: Politikaların gerçekleştirimi sağlanmadan önce politikalar bir yerde toplanmalıdır. KAoS ve Ponder'da bu merkezi olarak yerine getirilmektedir. Rei, WSPL ve XACML'de politika gerçekleştirimi merkezi olurken politikalar dağıtık olarak ağdan toplanmaktadır. Protune'da ise politikaların gerçekleştirimi dağıtık olmaktadır.

Uzlaşma: Sadece Protune politika dili uzlaşma sağlamaktadır.

Sonuç Şekli: Politika gerçekleştiriminin sonucunun istekte bulunan varlığa döndürülmesi gerekmektedir. KAoS ve Ponder'da bu yanıt istek *alındı* ya da *alınmadı* şeklindedir. WSPL ve XACML bu iki yanıtın yanı sıra uygulanabilir kural ya da politika olmadığını belirten *uygulanamaz* ve işlem sırasında hata oluştuğunu bildiren *belirsiz* yanıtlarınıda içermektedir. Protune, gelişmiş açıklama yeteneklerine izin vermektedir. İstekte bulunan varlık, isteğinin neden gerçekleşmediğinin yanı sıra isteğinin gerçekleşmesi için hangi adımları gerçekleştirmesi gerektiğinide sorabilmektedir. Rei politika dilinde *alındı* ve *alınmadı* yanıtlarının yanı sıra istekte bulunan varlık, zorunluluk politikalarında, zorunluluğu gerçekleştirmesinin ve gerçekleştirmemesinin etkilerini what-if sorgulamaları ile karşılaştırarak zorunluluğu tamamlayıp tamamlamamaya karar verebilir.

Genişletilebilirlik: Kullanıcı kendi gereksinimlerine göre dili uyarlayabilir. WSPL dışında bütün politika dilleri genişletilebilirliğe izin vermektedir.

Çizelge 4.2 Anlamsal Web politika dilleri karşılaştırması.

	KAoS	**Rei**	**Ponder**	**Protune**	**XACML**	**WSPL**
İyi Tanımlanmış Anlamsallık	Var	Var	Yok	Var	Yok	Yok
Altyapıdaki Biçim	DL	Deontik Mantık, Mantık Programlama, DL	Nesneye Dayalı	Mantık Programlama	-	-
Eylemin Yürütülmesi	Yok	Yok	Var (sistem özelliklerine erişirken)	Var	Var	Var
Yetki Aktarımı	Yok	Var	Var	Var	Yok	Yok
Gerçekleştirim Türü	Merkezi	Dağıtık Politikalar, Merkezi Gerçekleştirim	Merkezi	Dağıtık	Dağıtık Politikalar, Merkezi Gerçekleştirim	Dağıtık Politikalar, Merkezi Gerçekleştirim
Uzlaşma	Yok	Yok	Yok	Var	Yok	Yok
Sonuç Şekli	İzin/Yasak	İzin/Yasak*	İzin/Yasak	Açıklamalar	İzin/Yasak, Uygulanamaz, Belirsiz	İzin/Yasak, Uygulanamaz, Belirsiz
Genişletilebilirlik	Var	Var	Var	Var	Var	Yok

* Zorunluluk için "What-If" sorguları mevcut.

5. ONTOLOJİ TABANLI ERİŞİM DENETİMİ (OBAC)

Dizgenin davranış biçimini belirten politikaların tanımlanmasında ontoloji dillerinden yararlanılmaktadır ve ontolojiler kullanılarak iki konu modellenmektedir: (i) erişim denetim düzeneği (ii) kaynağın ve bu kaynağa erişmek isteyen varlığın bilgisi. Oluşturulan bu yapı Ontoloji Tabanlı Erişim Denetimi (Ontology Based Access Control - OBAC) modeli olarak tanımlanmaktadır. OBAC'te; erişilecek *nesne (object),* bu nesne üzerinde gerçekleştirilebilecek *eylemler (actions)* ve bu eylemlerin hangi *koşullar (conditions)* altında gerçekleştirilebileceğini belirten kısıtlar ontolojik olarak tanımlanmaktadır. Böylelikle, anlamsal olarak bütünlüğü olan parçaların yönetilmesi sağlanmaktadır. RBAC ve ABAC modellerinde erişilecek kaynak ile ilgili olarak herhangi bir üst veri bilgisi bulunmamaktadır. RBAC modelinde erişim gerçekleştirmek isteyen varlığın bilgisi temel alınırken, ABAC modelinde varlığın üzerinde işlem yapmak istediği nesneler ile ilgili öznitelikler temel alınmaktadır. OBAC modelinde ise erişilecek kaynak ve bu kaynağa erişecek varlık ile ilgili üst veriler anlamsal olarak oluşturulmakta, politikalar da bu üst veriler temel alınarak yaratılmaktadır.

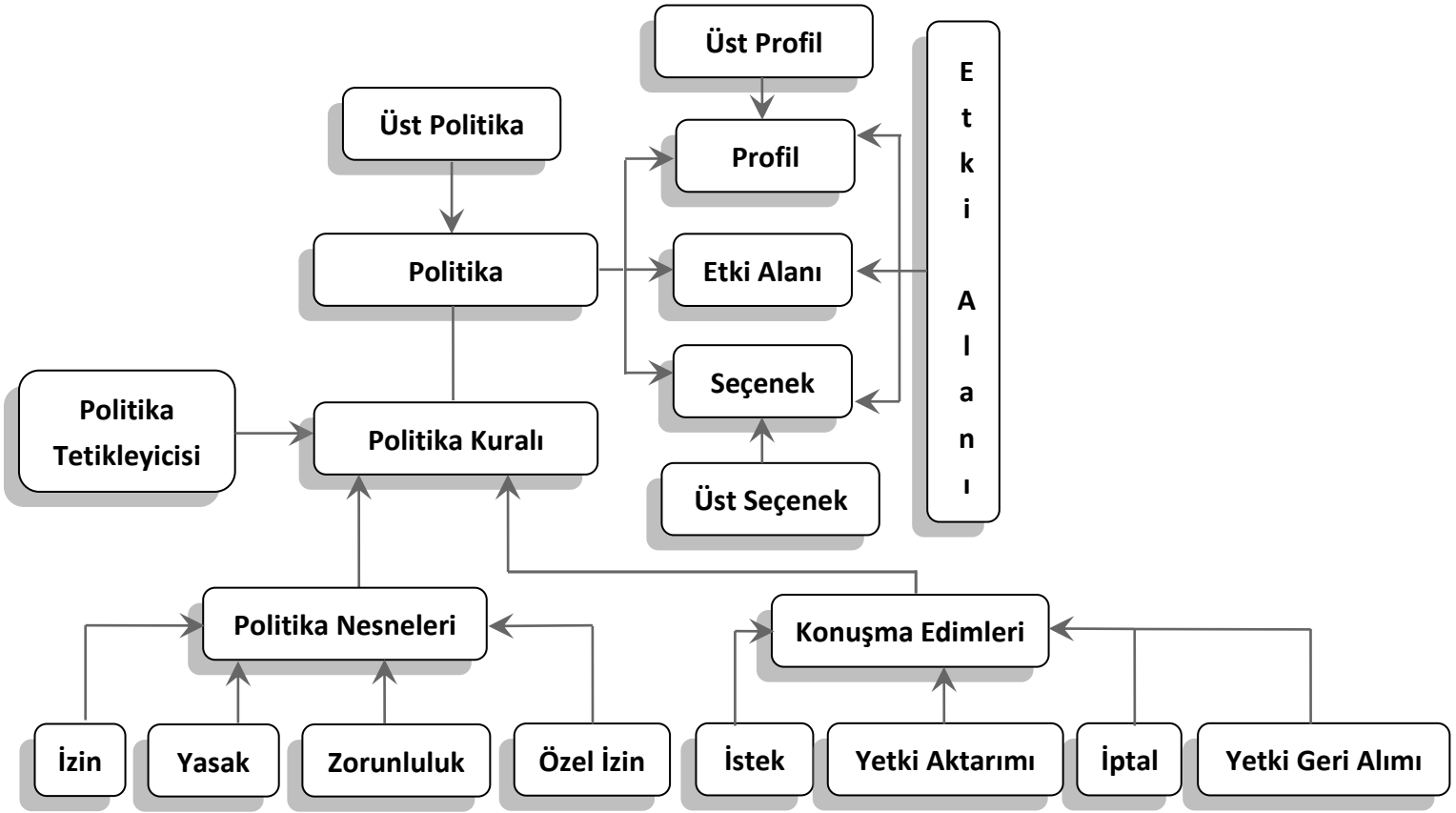
5.1 OBAC Modeli

Anlamsal Web teknolojilerine dayanan politika dilleri, politikaların çok çeşitli etki alanı verileri üzerinde tanımlanmasına izin vermekte ve aynı bilgi modelini kullanmayan katılımcılar arasında ortak anlamı

desteklemektedir (Finin et al., 2008). Bu tezde de, farklı profiller ve bir profilin çeşitli seçenekleri yukarıda verilen tanımda yer alan çok çeşitli etki alanı verilerini oluşturmaktadır. Profillerini oluşturan ve güncelleyen kullanıcılar ise dizgenin katılımcılarını oluşturmaktadır.

Kişiselleştirme kişiselleştirilmiş bilgi sistemlerinde bir gereksinimdir (Gauch et al., 2007) ve ayrıca kullanıcı beklentilerini karşılamaktadır (Rich, 1999). Bu nedenle, kişiselleştirme, ilgisiz bilgi süzülerek ve kullanıcının benzer ilgileri ile ilgili ek bilgi tanımlanarak sağlanabilir (Gauch et al., 2007). Kişiselleştirilmiş bir bilgi dizgesinde kullanıcılar seçimlerini tanımlamakta ve gereksinimleri doğrultusunda etki alanını sorgulamaktadır. Kullanıcı bağlamını ve kişiselleştirilmiş bilgi sistemini kaydetmek için kullanıcı profilleme kullanılmaktadır (Katifori et al., 2007).

OBAC modelinde tanımlanan politika ontolojileri ve politika kavramları arasındaki ilişkiler Şekil 5.1'de yer almaktadır. Sistemin daha iyi bir kişiselleştirme için kullanıcı gereksinimlerine yönelik olarak davranabilmesi amacıyla etki alanı, profil ve seçenek politikaları olmak üzere üç çeşit politika tanımlanmaktadır. Etki alanı verisi; etki alanı, profil ve seçenek tabanlı politika ontolojilerini oluşturmak için kullanılmaktadır. Politika ontolojileri üst politika olarak aldığımız Rei politika dili temel alınarak yaratılmaktadır. Ayrıca, profil tabanlı politikalar üst profil ontolojisi temel alınarak, seçenek tabanlı politikalar ise üst seçenek ontolojisi temel alınarak yaratılmaktadır.



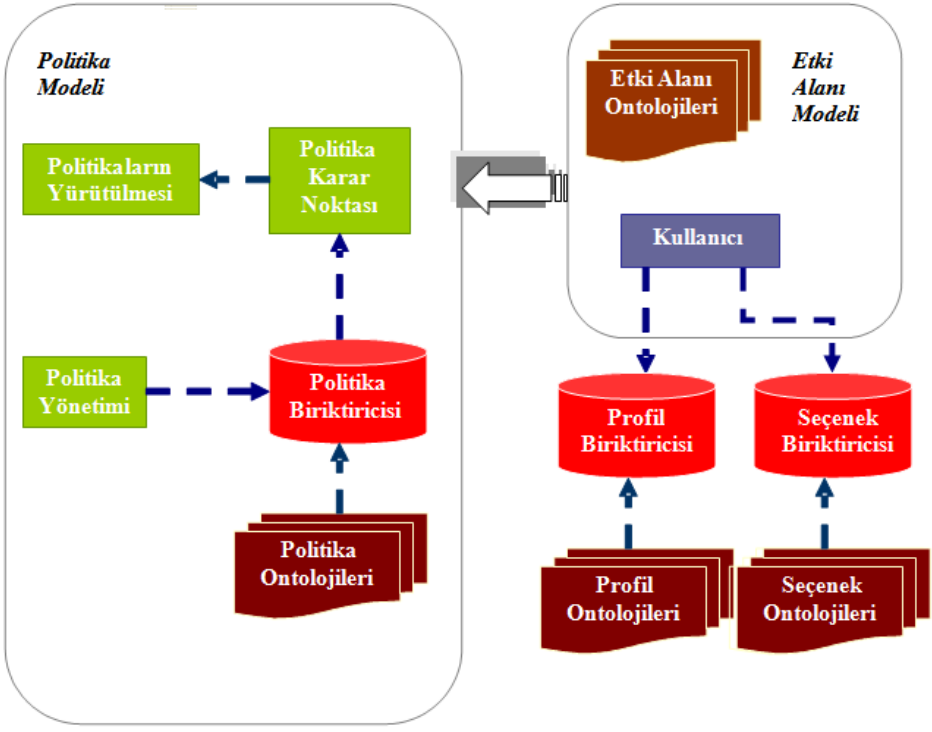
Şekil 5.1 OBAC politika ontolojileri ve politika kavramları arasındaki ilişki.

Politika, politika nesnelerinden oluşan politika kurallarından oluşmaktadır. Bu tezde; etki alanı, profil ve kişiselleştirmeyi sağlamak için bu profillerin seçenekleri üzerine politika kuralları tanımlanmaktadır. OBAC modelinde politika nesneleri olarak izin, yasak, zorunluluk ve özel izin tanımlanmaktadır. *İzin*, veriye ya da sunuya erişenin neleri yapabileceğini; *yasak*, neleri yapamayacağını; *zorunluluk*, neleri yapması gerektiğini; *özel izin* ise neleri yapmasına gerek kalmadığını belirten durumdur. Modelde yer alan politika tetikleyicisi politika kurallarının yürütülmesini sağlamaktadır. Konuşma edimleri olarakda istek, yetki aktarımı, iptal ve yetki geri alımı tanımlanmaktadır.

Dizgedeki kullanıcılar kendi profillerini ve seçeneklerini oluşturmaktadır. Politika yöneticisi tarafından oluşturulan politikalar da politika biriktiricisinde saklanmaktadır. Şekil 5.2'de görüldüğü gibi politikaların yürütülmesi için, her bir etki alanı, profil ve seçenek için tanımlanan politika kuralları politika karar noktasından geçip yürütülmektedir. Profil ve seçenek politikaları oluşturulurken sırası ile profil ve seçenek ontolojileri temel alınmaktadır. Profil politikası, üst profil (metaprofile) ontolojisi ve onu temel alan profil (profile) ontolojisi kullanılarak, seçenek politikası ise PMO (Preference Meta Ontology), PVD (Preference View Domain) ve PVD_Domain ontolojileri kullanılarak oluşturulmaktadır. Bu ontolojiler 6.2.2 ve 6.2.3 bölümlerinde anlatılmaktadır.

OBAC modelinde, kişiselleştirilmiş bir erişim denetimi için etki

alanı ve profil politikaları kullanılarak kullanıcı seçenekleri kısıtlanmaktadır. Kullanıcı, seçeneklerini olabilecek seçenekler arasından yapmaktadır. Böylelikle, seçenekler kısıtlandırılarak en iyi sorgu sonuçlarına ulaşılabilecektir. Örneğin, eğer bir turist evcil hayvanlara izin verilen bir odada kalmak istiyorsa, otel politikasının yürütülmesinden sonra kullanıcıya evcil hayvanlara izin verilen otellerin isimleri listelenecektir. Politika ve seçenek arasındaki ontolojik ilişkinin sağlanabilmesi için politika eylemleri seçenekleri temel almaktadır.



Şekil 5.2 Politika, Profil, Seçenek ve Etki Alanı ilişkileri.

Ontolojiler genellikle veri biriktiricilerinde saklanmaktadırlar. OBAC mimarisi, kullanıcıların bu veri biriktiricilerine doğrudan erişimlerine izin vermemektedir. Eğer bir kullanıcı ontolojilere erişmek isterse, OBAC mimarisinin o kullanıcının profiline verdiği erişim izinleri doğrultusunda eylemlerini gerçekleştirebilir. Buradaki temel nokta, OBAC modelinin erişim denetiminin sağlanmasında gerçekleştirilen işlemler için kullanıcıya yönelik olarak soyut bir engel sağlamasıdır.

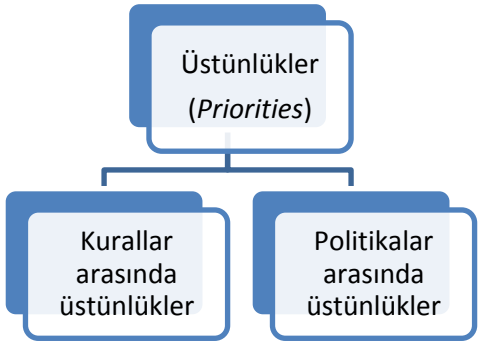
5.1.1 OBAC'te Çelişkilerin çözümü

Politika bir sistemin hangi koşullar altında davranacağını tanımlamak için kullanılmaktadır. Politikalar tanımlanırken yapılmış olan hatalar, eksiklikler ve çelişen gereksinimler politika çelişkilerini meydana getirmektedir (Lupu and Sloman, 1999). Bir nesne için birden fazla politika tanımlanması çelişkinin ortaya çıkması olasılığını arttırmaktadır. Aynı hedefte aynı eylem için farklı politikaların bulunması politika çelişkilerinin ortaya çıkmasına neden olmaktadır. Böyle bir durumda politikaların yeniden yazılması istenilen bir çözüm biçimi değildir. Politika çelişkilerinin çözümlenebilmesi için üstünlükler ve öncelik ilişkileri tanımlanmalıdır (Tonti et al., 2003).

Çelişkilerin çözümlenmesinde Rei politika dilinin üst-politika (*meta-policy*) tanımlamaları kullanılmaktadır. Üst-politikalar politikaların nasıl yorumlandığı ve çelişkilerin devingen olarak nasıl çözümlendiği ile ilgili politikalardır.

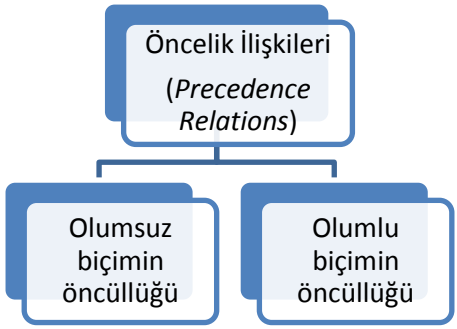
Politika çelişkileri iki şekilde ortaya çıkmaktadır: politikaların çelişmesi ve kuralların çelişmesi. Üstünlükler ve öncelik ilişkileri tanımlanarak bu çelişkilerin çözümlenmesi sırası ile Şekil 5.3 ve Şekil 5.4'de gösterilmiştir.

Üstünlükler ile çelişkinin çözümünde kurallar ve politikalar arasında üstünlükler tanımlanmaktadır. Örneğin; "*A1 kuralı B1 kuralının üzerine yazar.*" ya da "*A1 politikası B1 politikasının üzerine yazar.*"



Şekil 5.3 Çelişkilerin üstünlükler ile çözümü.

Öncelik ilişkileri ile çelişkilerin çözümünde olumsuz biçimin (*negative modality*) önceliği ve olumlu biçimin önceliği (*positive modality*) tanımlanmaktadır. Olumsuz biçimin önceliğinde yasağın izin üzerine önceliği vardır ve özel izin zorunluluktan daha güçlüdür. Olumlu biçimin önceliğinde ise izinin yasak üzerine önceliği vardır ve zorunluluk özel izinden daha güçlüdür.



Şekil 5.4 Çelişkilerin öncelik ilişkileri ile çözümü.

Kuralların çelişmesinin çözümünde Rei politika dilinin MetaPolicy ontolojisinden `ruleOfGreaterPriority` ve `ruleOfLesser Priority` nesne özellikleri kullanılmaktadır. Kuralların çelişmesinin çözümü deontik politika tanımı yapılırken aşağıdaki iki algoritma kullanılmaktadır. Bunlardan birincisi `deontic:Permission` örneği yaratılırken kullanılan algoritmadır:

deonticPermissionConflictResolution *(String deonticPolicyAction, OntClass deonticPolicyCls, OntModel tourismModelPolicy, String Source, String policyOntologyURI, String deonticPolicyName, String deonticPolicyType)*

for *each individual of prohibitionClass of tourismModelPolicy*

if *(deonticProhibitionActionProperty==deonticPermissionActionProperty)*

print("Oops! There is a conflict..");

String choice=choose("PermOverridesPro"|| "ProOverridesPer")

if *(choice=="PermOverridesPro")*

set metaPolicy_ruleOfGreaterPriority objectProperty deonticPermission

set metaPolicy_ruleOfLesserPriority objectProperty deonticProhibition

else if *(choice=="ProOverridesPer")*

set metaPolicy_ruleOfGreaterPriority objectProperty deonticProhibition
set metaPolicy_ruleOfLesserPriority objectProperty deonticPermission

İkinci algoritma ise `deontic:Prohibition` örneği yaratılırken kullanılan algoritmadır:

deonticProhibitionConflictResolution *(String deonticPolicyAction, OntClass deonticPolicyCls, OntModel tourismModelPolicy, String Source, String policyOntologyURI, String deonticPolicyName, String deonticPolicyType)*

for *each individual of permissionClass of tourismModelPolicy*
if *(deonticProhibitionActionProperty==deonticPermissionActionProperty)*
print("Oops! There is a conflict..");
String choice=choose("PermOverridesPro" || "ProOverridesPer")
if *(choice=="PermOverridesPro")*
set metaPolicy_ruleOfGreaterPriority objectProperty deonticPermission
set metaPolicy_ruleOfLesserPriority objectProperty deonticProhibition
else if *(choice=="ProOverridesPer")*
set metaPolicy_ruleOfGreaterPriority objectProperty deonticProhibition
set metaPolicy_ruleOfLesserPriority objectProperty deonticPermission

Politikaların çelişmesinin çözümünde ise Rei politika dilinin MetaPolicy ontolojisinden `policyOfGreaterPriority` ve `policyOfLesserPriority` nesne özellikleri kullanılmaktadır. Kuralların çelişmesinin çözümü politika tanımı yapılırken aşağıdaki algoritma kullanılmaktadır:

conflictResolutionPolicy *(String policyGrants, OntModel tourismModelPolicy, String Source, String policyOntologyURI, String policyName)*

for *the new individual of policyClass of tourismModelPolicy*

```
Individual policyIndv=policyName of tourismModelPolicy
RDFNode deonticPolicyNode=policyDeonticProperty of  policyGrants
RDFNode policyActionNode1=policyActionProperty of policyIndv
if (deonticPolicyNode.contains("Permission"))
    for each individual of policyClass of tourismModelPolicy
     policyClassIndv=individual of policyClass
     policyActionNode2=policyActionProperty of policyClassIndv
     Individual policyClassGrantsProperty=
                        get policyGrantsPropertyIndividual of  tourismModelPolicy
    RDFNode grantsDeonticPolicyNode=get policyDeonticProperty of
                        policyClassGrantsProperty
    if (policyActionNode1==policyActionNode2 &&
                              grantsDeonticPolicyNode.contains("Prohibition"))
     print("Oops! There is a conflict..");
     String choice=choose("PermOverridesPro" || "ProOverridesPer")
     if (choice==PermOverridesPro)
      set metaPolicy_policyOfGreaterPriority objectProperty deonticPermission
      set metaPolicy_policyOfLesserPriority objectProperty deonticProhibition
    else if (choice==ProOverridesPer)
     set metaPolicy_policyOfGreaterPriority objectProperty deonticProhibition
    set metaPolicy_policyOfLesserPriority objectProperty deonticPermission
else if (deonticPolicyNode.contains("Prohibition"))
    for each individual of policyClass of tourismModelPolicy
     policyClassIndv=individual of policyClass
     policyActionNode2=policyActionProperty of policyClassIndv
     Individual policyClassGrantsProperty=
                        get policyGrantsPropertyIndividual of  tourismModelPolicy
    RDFNode grantsDeonticPolicyNode=get policyDeonticProperty of
                        policyClassGrantsProperty
    if (policyActionNode1==policyActionNode2 &&
                              grantsDeonticPolicyNode.contains("Permission"))
```

```
print("Oops! There is a conflict..");
String choice=choose("PermOverridesPro" || "ProOverridesPer")
if (choice==PermOverridesPro)
  set metaPolicy_policyOfGreaterPriority objectProperty deonticPermission
  set metaPolicy_policyOfLesserPriority objectProperty deonticProhibition
else if (choice==ProOverridesPer)
  set metaPolicy_policyOfGreaterPriority objectProperty deonticProhibition
  set metaPolicy_policyOfLesserPriority objectProperty deonticPermission
```

Eğer bir çelişki meydana geliyorsa, son karar belirtilen üstünlük politikası tarafından verilmektedir. Örneğin, "*Konferans katılımcıları dağ manzaralı odalarda kalabilirler.*" şeklindeki bir otel politika kuralına karşılık kullanıcının yapmış olduğu oda manzarası seçeneği "`deniz manzaralı`" ise bu durumda bir çelişki meydana gelmiş olacaktır. Kullanıcı profili `turist` ya da `konferans katılımcısı` şeklinde olabilir. Eğer kullanıcı `konferans katılımcısı` profilinde işlem yaparsa son karar bir yasak şeklinde olacak, ancak eğer `turist` profilinde işlem yaparsa oda manzarası seçeneğinde herhangi bir kısıtlama ile karşılaşmayacaktır.

Çelişkiler; *seçenek-seçenek, seçenek-politika* ve *politika-politika* arasında olmak üzere üç açıdan incelenmektedir. Kullanıcı seçenekleri zaman içerisinde ya da koşullara bağlı olarak değişebilir. Kullanıcının birbiri ile çelişen uyumsuz seçenekler yapması sonucunda *seçenek-seçenek* çelişkisi meydana gelmektedir. Örneğin, oda seçeneği yapmakta olan bir kullanıcı eğer `hasInternetAccess` seçeneğini `false` ve `wirelessConnection` seçeneğini `true` seçtiğinde bir çelişki meydana

gelecektir. `wirelessConnection` seçeneğinin `hasInternetAccess` seçeneğini geçersiz kıldığını (*override*) belirten bir üst-politika kuralı tanımlanmıştır.

Aşağıdaki örnekte öncelikle `metapolicy:rulePriority` tanımı yapılmaktadır. Daha sonra, `wirelessConnection` izininin daha yüksek öncelikte olduğunu belirten `metapolicy: ruleOfGreaterPriority` tanımı yer almaktadır.

```
<metapolicy:rulePriority
     rdf:ID="WirelessConnectionPreferenceOverridesInternet
            AccessPreference">
  <metapolicy:ruleOfGreaterPriority>
    <deontic:Permission
            rdf:ID="Permission_WirelessConnectionPreference">
```

Aşağıda, `deontic:constraint` ile `wirelessConnection` izini için `any_Guest` kısıt tanımı yapılmaktadır. Burada Profil ontolojisinde yer alan `Guest` türünde bir `any_Guest` tanımlanmaktadır.

```
      <deontic:constraint>
        <constraint:SimpleConstraint rdf:ID="any_Guest">
          <constraint:predicate>
            <owl:Thing
            rdf:about="http://www.w3.org/1999/02/22-rdf-
            syntax-ns#type"/>
          </constraint:predicate>
          <policy:desc
            rdf:datatype="http://www.w3.org/2001/XMLSchema#
                        string">Any guest.
        </policy:desc>
          <constraint:subject>
            <entity:Variable rdf:ID="Guest">
              <policy:desc
                rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#string">A guest of the hotel.
              </policy:desc>
            </entity:Variable>
```

```
    </constraint:subject>
    <constraint:object
      rdf:resource="http://efe.ege.edu.tr/~odo/
      Ontology/Profile.owl#Guest"/>
  </constraint:SimpleConstraint>
</deontic:constraint>
```

Aşağıda tanımlanmış olan `deontic:action` ile `wirelessConnection` izininin eylemi olan `WirelessConnection` belirtilmektedir. Bu eylem, kısıtı `any_Guest` olan bir `Guest` tarafından PVD_Domain ontolojisinde yer alan `HiltonParisGuestroom`'da gerçekleştirilmektedir.

```
<deontic:action>
  <WirelessConnection
     rdf:ID="WirelessConnectionPreference">
     <action:location
           rdf:resource="http://efe.ege.edu.tr/~ozgucan/
           PhD/Ontology/PVD_Domain.owl#HiltonParisGuestr
           oom_wirelessConnection"/>
       <action:precondition rdf:resource="#any_Guest"/>
       <policy:desc
           rdf:datatype="http://www.w3.org/2001/
           XMLSchema#string">Wireless Connection
           preference for Hilton Paris is true.
     </policy:desc>
    <action:actor rdf:resource="#Guest"/>
  </WirelessConnection>
</deontic:action>
```

İzinin aktörü olan `Guest` aşağıdağıda tanımlanmış olan `deontic:actor` belirtilmektedir. Daha sonra, `wirelessConnection` izini ve `metapolicy:ruleOfGreaterPriority` tanımı sonlandırılmaktadır.

```
<deontic:actor>
  <owl:Thing
     rdf:about="http://efe.ege.edu.tr/~odo/Ontology/
     Profile.owl#Guest"/>
</deontic:actor>
```

```
      </deontic:Permission>
    </metapolicy:ruleOfGreaterPriority>
```

`Permission_InternetAccessPreference` ile internet erişim iznini belirtilmektedir. Bu izin daha düşük önceliğe sahip olduğundan `metapolicy:ruleOfLesserPriority` tanımı içerisinde yer almaktadır. Bu izin, `Guest` aktörü tarafından `any_Guest` kısıtı ile `HiltonParisGuestroom` için tanımlanmaktadır.

```
    <metapolicy:ruleOfLesserPriority>
      <deontic:Permission
            rdf:ID="Permission_InternetAccessPreference">
        <deontic:action>
          <AccessingInternet rdf:ID="InternetAccessPreference">
            <action:location
            rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
              Ontology/PVD_Domain.owl#HiltonParisGuestroom_
              internetAccess"/>
            <action:precondition rdf:resource="#any_Guest"/>
            <policy:desc
            rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string">Internet access preference for Hilton
              Paris is false.
          </policy:desc>
            <action:actor rdf:resource="#Guest"/>
          </AccessingInternet>
        </deontic:action>
        <deontic:actor
           rdf:resource="http://efe.ege.edu.tr/~odo/Ontology/
             Profile.owl#Guest"/>
        <deontic:constraint rdf:resource="#any_Guest"/>
      </deontic:Permission>
    </metapolicy:ruleOfLesserPriority>
  </metapolicy:rulePriority>
```

Seçenek-politika çelişkisinde; aynı seçenekler üzerine belirtilmiş farklı politikalar üzerine çalışılmıştır. Örneğin, eğer yarım pansiyon seçimi yapmış olan konuklar için iki farklı politika kuralı tanımlanmış ise:

Yarım pansiyon seçen konuklar açık büfeyi kullanabilirler. (`Permission_Snackbar`)

Yarım pansiyon seçen konuklar açık büfeyi kullanamazlar. (`Prohibition_Snackbar`)

bu *seçenek-politika* çelişkisinin çözülmesi için, `Prohibition_Snackbar` kuralının `Permission_SnackBar` kuralını geçersiz kıldığı, aşağıdaki üst-politika kuralı tanımlanmıştır:

```
<metapolicy:rulePriority
       rdf:ID="ProhOverridesPerm_SnackBar">
  <metapolicy:ruleOfLesserPriority
       rdf:resource="#Permission_SnackBar"/>
  <metapolicy:ruleOfGreaterPriority
       rdf:resource="#Prohibition_SnackBar"/>
</metapolicy:rulePriority>
```

Yukarıdaki kuralda, yüksek kural önceliğini tanımlayan `metapolicy:ruleOfGreaterPriority` özelliğine `Prohibition_Snackbar` kuralı, düşük kural önceliğini tanımlayan `metapolicy:ruleOfLesserPriority` özelliğine `Permission_SnackBar` kuralı atanmaktadır.

Modelde ortaya çıkabilecek *politika-politika* çelişkileri için de politikalar arasında öncelikler belirlenmektedir. Bu nedenle, çelişen politika ontolojileri için üst-politika kuralları tanımlanmaktadır. Aşağıdaki üst-politika kuralı bir *politika-politika* çelişkisine örnektir:

Şirket çalışanı politikasının Turist politikasından daha büyük bir önceliği vardır.

```
<metapolicy:RulePriority
   rdf:ID="CompanyEmployeePolicyOverridesTouristPolicy">
  <metapolicy:policyOfLesserPriority
       rdf:resource="#TouristPolicy"/>
      <metapolicy:policyOfGreaterPriority
```

```
        rdf:resource="#CompanyEmployeePolicy"/>
    </metapolicy:RulePriority>
```

Bu kuralda, yüksek politika önceliğini tanımlayan `metapolicy:policyOfGreaterPriority` özelliğine `Company EmployeePolicy` politikası, düşük politika önceliğini tanımlayan `metapolicy:policyOfLesserPriority` özelliğine `Tourist Policy` politikası atanmaktadır.

5.1.2 OBAC Konuşma edimleri

Konuşma edimleri farklı varlıklar arasındaki etkileşimlerin modellenmesi için kullanılmaktadır. Rei konuşma edimleri FIPA[1] (Foundation for Intelligent Physical Agents) tanımlamalarını temel almaktadır (Kagal, 2002). Etmenlerin eğer ilgili yetkileri varsa konuşma edimlerini kullanabilirler.

Ontoloji Tabanlı Erişim Denetimi'nde konuşma edimleri diğer OBAC bileşenlerinden bağımsız bir biçimde, ayrı bir ontoloji olarak oluşturulmaktadır. 4 konuşma edimi tanımlanmaktadır:

- *İstek (Request):* Eylem ile ilgili olarak istekte bulunulmasıdır.
- *Yetki aktarımı (Delegation):* Eylem ile ilgili yetkinin istekte bulunan varlığa verilmesidir.
- *İptal (Cancel):* İsteğin iptal edilmesidir.
- *Yetkinin geri alımı (Revocation):* Verilen yetkinin geri

[1] FIPA, http://www.fipa.org/ .

alınmasıdır.

Bu bölümde, yetki aktarımı verilmesi ve yetkinin geri alınması ile ilgili iki örnek verilmektedir. Aşağıdaki örnekte *Şirket Çalışanı*, *Yönetici*'den *açık büfeyi kullanma izni* ile ilgili bir istekte bulunmakta ve *Yönetici*, *Şirket Çalışanı*'na *açık büfeyi kullanma izni* ile ilgili yetki aktarımını gerçekleştirmektedir.

Örnekte, öncelikle açık büfeyi kullanma izini olan `Permission_UsingSnackbar` tanımı yapılmaktadır. Bu izin tanımı Profil politika ontolojisinde yapılmış olan `Using_Snackbar` eylemini kullanmaktadır.

```
<deontic:Permission rdf:ID="Permission_UsingSnackbar">
     <deontic:action
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/ProfilePolicy.owl#Using_Snackbar"/>
</deontic:Permission>
```

Aşağıda, şirket çalışanının yöneticiye yapmış olduğu `RequestCompanyEmployeeToManager` yetki isteği tanımı yer almaktadır. Bu istek tanımı, isteği gönderen şirket çalışanı değeri `action:sender` özelliğine Profil ontolojisinde yer alan `CompanyEmployee` değeri, isteği alan yönetici ise `action:receiver` özelliğine yine Profil ontolojisinde yer alan `Manager` değeri aktarılarak yapılmaktadır. İstek eyleminin içeriğini belirten `UsingSnackbar` tanımı Profil politikası ontolojisinden gelmektedir.

```
<action:Request rdf:ID="RequestCompanyEmployeeToManager">
     <action:sender
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/Profile.owl#CompanyEmployee"/>
```

```
      <action:content
            rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
            Ontology/ProfilePolicy.owl#Using_Snackbar"/>
      <action:receiver
            rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
            Ontology/Profile.owl#Manager"/>
</action:Request>
```

Yöneticinin şirket çalışanının yetkiyi aktarmasını gösteren `DelegationManagerToCompanyEmployee` tanımı aşağıda yer almaktadır. Yetki aktarımı tanımı, yetkiyi gönderen yönetici değeri `action:sender` özelliğine Profil ontolojisinde yer alan `Manager` değeri, yetkiyi alan şirket çalışanı ise `action:receiver` özelliğine yine Profil ontolojisinde yer alan `CompanyEmployee` değeri aktarılarak yapılmaktadır. Yetki aktarımı eyleminin içeriğini belirten `UsingSnackbar` tanımı Profil politikası ontolojisinden gelmektedir.

```
<action:Delegation rdf:ID="DelegationManagerToCompanyEmployee">
      <action:content
            rdf:resource="file:/C:/OBAC/SpeechAct/SpeechActs_DelReq
            .owl#Permission_UsingSnackbar"/>
      <action:sender
            rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
            Ontology/Profile.owl#Manager"/>
      <action:receiver
            rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
            Ontology/Profile.owl#CompanyEmployee"/>
</action:Delegation>
```

Aşağıdaki örnekte ise yukarıdaki örnekte verilmiş olan yetkinin geri alımı gerçekleştirilmektedir. *Yönetici*, *Şirket Çalışanı*'na verdiği *açık büfeyi kullanma iznini* geri alıp iptal etmektedir.

Profil politika ontolojisinde yapılmış olan `Using_Snackbar` eylemini kullanan açık büfeyi kullanma izini `Permission_`

UsingSnackbar tanımı aşağıda verilmektedir.

```
<deontic:Permission rdf:ID="Permission_UsingSnackbar">
       <deontic:action
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/ProfilePolicy.owl#Using_Snackbar"/>
</deontic:Permission>
```

Şirket çalışanının yöneticiye gönderdiği isteği iptal eden Cancel_CompanyEmployeeToManager tanımı aşağıda yer almaktadır. İptal tanımı, bu isteği gönderen şirket çalışanı değeri action:sender özelliğine Profil ontolojisinde yer alan CompanyEmployee değeri, bu isteği alan yönetici ise action:receiver özelliğine yine Profil ontolojisinde yer alan Manager değeri aktarılarak yapılmaktadır. İptal eyleminin içeriğini belirten Using_Snackbar tanımı Profil politikası ontolojisinden gelmektedir.

```
<action:Cancel rdf:ID="Cancel_CompanyEmployeeToManager">
       <action:content
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/ProfilePolicy.owl#Using_Snackbar"/>
       <action:sender
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/Profile.owl#CompanyEmployee"/>
       <action:receiver
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/Profile.owl#Manager"/>
</action:Cancel>
```

Aşağıda, yöneticinin şirket çalışanına gönderdiği yetkiyi geri alması olan Revocation_ManagerToCompanyEmployee yetki geri alımı tanımı yer almaktadır. Yetki geri alımı tanımı, göndermiş olduğu

yetkiyi geri alan yönetici değeri `action:sender` özelliğine Profil ontolojisinde yer alan `Manager` değeri, yetkinin geri alındığı şirket çalışanı ise `action:receiver` özelliğine yine Profil ontolojisinde yer alan `CompanyEmployee` değeri aktarılarak yapılmaktadır. Yetkinin geri alınması eyleminin içeriğini belirten `UsingSnackbar` tanımı Profil politikası ontolojisinden gelmektedir.

```
<action:Revocation rdf:ID="Revocation_ManagerToCompanyEmployee">
      <action:sender
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/Profile.owl#Manager"/>
      <action:receiver
          rdf:resource="http://efe.ege.edu.tr/~ozgucan/PhD/
          Ontology/Profile.owl#CompanyEmployee"/>
      <action:content
          rdf:resource="file:/C:/OBAC/SpeechAct/SpeechActs_DelReq
          .owl#Permission_UsingSnackbar"/>
</action:Revocation>
```

5.2 RBAC, ABAC ve OBAC Karşılaştırması

Anlamsal Web teknolojilerine dayanan politika dilleri, politikaların heterojen etki alanı verileri üzerinde tanımlanmasına izin vermekte ve aynı bilgi modelini kullanmayan katılımcılar arasında ortak anlamı desteklemektedir (Finin et al., 2008). Son yıllarda yapılan erişim kontrol çalışmalarında iki paralel konu ele alınmaktadır (Finin et al., 2008): gerçek dünya uygulama etki alanlarının politika gereksinimlerini karşılamaya yönelik erişim kontrol modellerinin geliştirilmesi ve erişim kontrolü için politika dillerinin geliştirilmesi. Bu iki paralel konunun, erişim denetimi ve politika dilleri, güvenlik altyapısının gelişimini sağlamak için görevdeşlik yaratması gerektiği düşünülmektedir. OBAC

modelinde politika yönetimi ile erişim denetimi birleştirilmektedir. Mevcut erişim denetimlerinde olmayan bu nokta, bu tezin en önemli katkılarından biridir.

OBAC modelinde önerilen çözüm, erişim denetim kavramlarının politika dilleri tarafından desteklenen modeller içerisinde doğal bir şekilde belirtilmesi ve modellerin ötesindeki detayların ise politika dili içerisinde ayrıca belirtilmesine izin verilmesidir.

RBAC modelinde, güvenlik politikası izinleri, kullanıcıya değil rollere verilmektedir. Kullanıcı, izinlerini sistemde tanımlı olan rollerine göre elde etmektedir. Böylece, kullanıcı rolleri ile ilişkili olan bütün izinlerini kalıt almaktadır ve bu rol sıradüzeni ayrıca politikaların tanımlanmasını da basitleştirmektedir (Cuppens and Miège, 2003). Ancak, RBAC modelinde, kullanıcı-rol ve izin-rol atamaları için yönetimsel işlemlerin gerekli olması, roller ve izinlerin sayısındaki üstel artışın kullanıcı-rol ve izin-rol atama işlemlerini masraflı bir hale getirmesi gibi bazı olumsuzluklara neden olmaktadır (Yuan and Tong, 2005a). RBAC modelinin yönetiminde, rollere kullanıcı ve izin atamalarının nasıl yapıldığı konusunda bir ayrıntı bulunmamaktadır (Finin et al., 2008). RBAC modelinde yer alan bu olumsuzlukları ortadan kaldırmak amacı ile OBAC modelinde profil tabanlı politika yaklaşımı önerilmektedir. Profiller, kullanıcı bilgilerinin saklandığı ve modellendiği ontolojik yapılar aracılığı ile ifade edilmektedir. Kullanıcı bilgileri, kullanıcının sahip olduğu bir takım öznitelik değerlerine göre ontolojiler aracılığı ile ifade edilmektedir. Kullanıcıların isteklerini ve bilgilerini

toplayabilmek, kullanıcıların bir sonraki işlemlerini tahmin edebilmek için önemlidir (Bursa ve Ünalır, 2008). Bu nedenle, kullanıcı bilgilerinin tutulması ve saklanması amacıyla kullanıcı profilleri oluşturulmuştur. Kullanıcı profillerinin modellenmesi, oluşturulması ve saklanması profil yönetimini oluşturmaktadır (Bursa ve Ünalır, 2008).

ABAC, özneler ve nesneler arasında izinleri doğrudan tanımlamak yerine, yetki tanımlamaları için onların özniteliklerini kullanmaktadır. ABAC yaklaşımında, rollerin tanımlanmasına gerek yoktur. Bu durum RBAC'ten farklılık göstermektedir. Bu açıdan bakıldığında, ABAC modelinin üstünlüğü rollerin tanımlanması ve yönetimi işlemlerini yok etmiş olmasıdır. Böylece, kullanıcı-rol ve izin-rol atamaları için gerekli olan yönetim görevleri de ortadan kalkmaktadır (Jrad and Aufare, 2007).

OBAC'te; erişilecek *nesne*, bu nesne üzerinde gerçekleştirilebilecek *eylemler* ve bu eylemlerin hangi *koşullar* altında gerçekleştirilebileceğini belirten kısıtlar ontolojik olarak tanımlanmaktadır. OBAC'in sunduğu bu model (Finin et al., 2008) çalışmasında ifade edilen gereksinimi karşılamaktadır. Böylelikle, anlamsal olarak bütünlüğü sağlanması gereken ontoloji tabanlı erişim denetimi ve politika yönetimi sağlanmaktadır. RBAC ve ABAC modellerinde erişilecek kaynak ile ilgili olarak herhangi bir üst veri bilgisi bulunmamaktadır. RBAC modelinde erişim gerçekleştirmek isteyen varlığın bilgisi temel alınırken, ABAC modelinde varlığın üzerinde işlem yapmak istediği nesneler ile ilgili öznitelikler temel alınmaktadır. OBAC modelinde ise erişilecek kaynak ve bu kaynağa

erişecek varlık ile ilgili üst veriler anlamsal olarak oluşturulmakta, politikalar da bu üst veriler temel alınarak yaratılmaktadır.

RBAC ve ABAC modellerinde bir dizgedeki bütün politika ayrıntılarının ifade edilmesini sağlayacak düzenek bulunmamaktadır. RBAC modelinde roller, isimlendirilmiş varlıklar olarak tanımlanabilmektedir. Bu durumun ontolojik olarak bir anlamsallığı bulunmamaktadır. Örneğin `genç akademisyen` tanımı yapılmak istendiğinde RBAC modelinde `genç` ve `akademisyen` tanımlamaları ayrı ayrı yapılabilirken bu iki tanımı birleştiren `genç akademisyen` durumu belirsiz kalmaktadır. OBAC modelinde ise `genç` ve `akademisyen` tanımları yapılırken, bu tanımlar sadece isimlendirilmiş birer varlık olarak alınmamaktadır. `Genç` tanımı için FOAF dosyasından alınan `yaş` bilgisi, örneğin; 18 ile 35 yaş arası olmak üzere sınırlandırılmaktadır. Aynı şekilde, `akademisyen` tanımı yapılırken ise `meslek` bilgisi temel alınmaktadır. `Genç akademisyen` tanımı, ontoloji modelindeki kesişim mantıksal operatörü ile sağlanmaktadır. RBAC modelinde, rol tanımı ontolojik olmadığından roller arasındaki ilişkiler tanımlanamamaktadır. OBAC modelinde ise bu durumun tanımını ontolojik olarak yapmak mümkündür.

RBAC ve ABAC modellerinde politika nesneleri sadece izin ve yasak şeklinde tanımlanırken OBAC modelinde izin, yasak, zorunluluk ve özel izin şeklinde olması OBAC modelinin RBAC ve ABAC modellerine olan bir diğer üstünlüğünü oluşturmaktadır. Buna ek olarak, RBAC ve ABAC modellerinde konuşma edimleri ile ilgili bir destek

bulunmamaktadır. OBAC modeli ise istek, yetki aktarımı, iptal ve yetki geri alımı olmak üzere 4 konuşma edimini de desteklemektedir.

Politika çelişkilerinin tespit edilip çözülmesi işlemleri de üst politika ontolojisi kullanılarak OBAC modelinde gerçekleştirilmektedir. RBAC ve ABAC modelleri politika çelişkileri ile ilgili herhangi bir çözüm sunmamaktadırlar.

RBAC modelini temel alan (Chen, 2008) çalışması, bilgiye erişimi ontoloji tabanlı erişim denetimi ile gerçekleştirilmektedir. Bu çalışma, bilgi içeriğini tanımlamak için bilgi kavramlarını ve ilişkilerini ontoloji tabanlı bir yaklaşım ile modellemektedir. Oluşturulan ontolojiler ürün etki alanını, rolleri ve süreçler arası ilişkileri tanımlamaktadır. Ancak, bu çalışmada sadece erişim denetim izinlerinin tanımlanması sağlanmakta ve politika çelişmelerinin çözümü gerçekleştirilmemektedir.

(Sun et al., 2007) ise RBAC modelini temel alan bir diğer ontoloji tabanlı erişim denetimi çalışmasıdır. Bu çalışmada, yetki kuralları için ortak bir ontoloji tanımlanmaktadır. Dizgeye kayıtlı kullanıcılar politikalar ve yetkiler kullanarak kendi kaynaklarının izinlerini belirlemektedirler. Dizgeye kayıtlı kullanıcılar arasında uyumun sağlanması ve dışardan gelen sorgulara cevap verebilmek için kullanıcıların kendi yerel yetkileri ile ortak ontoloji arasında eşlemeler yapılmaktadır. Bu çalışmada, erişim denetiminde sadece izinleri kullanmakta ve politika çelişmelerinin çözümünü gerçekleştirmemektedir.

RBAC, ABAC ve OBAC modellerinin karşılaştırması Çizelge 5.1'de yer almaktadır. Bu çizelgede, karşılaştırma, politika tanımlamada önemli bir yeri olan rol tanımı, politika nesneleri, konuşma edimleri ve politika çelişmesi çözümü açılarından yapılmaktadır.

Bu çizelgede; rol tanımı, modelin temel aldığı yapıyı belirtmektedir. RBAC modeli rol tabanlı, ABAC ise öznitelik tabanlı bir modeldir. OBAC modeli ise ontoloji tabanlı bir model olup roller yerine profiller kullanılmaktadır. RBAC ve ABAC, politika nesneleri olarak sadece izin ve yasak tanımlarken OBAC modelinde bu nesnelere ek olarak zorunluluk ve özel izin tanımlanmaktadır. Konuşma edimleri ile politika çelişmelerinin tespit ve çözümü sadece OBAC modeli tarafından desteklenmektedir.

Çizelge 5.1 RBAC, ABAC ve OBAC karşılaştırması.

	RBAC	**ABAC**	**OBAC**
Rol Tanımı	Rol tabanlı	Öznitelik tabanlı	Ontoloji tabanlı
Politika Nesneleri	İzin ve Yasak	İzin ve Yasak	İzin, Yasak, Zorunluluk ve Özel İzin
Konuşma Edimleri	Yok	Yok	İstek, Yetki aktarma, İptal ve Yetki geri alımı
Politika Çelişmesi Çözümü	Yok	Yok	Üstünlükler ve Öncelik İlişkileri

5.3 Ontoloji Tabanlı Erişim Denetimi ve Sunular

Bu tez kapsamında geliştirilmiş olan Ontoloji Tabanlı Erişim Denetimi, kullanıcıya aşağıda kısaca özetlenen hizmetleri sunmaktadır:

- İzin, yasak, zorunluluk ve özel izinden oluşan etki alanı, profil ve seçenek tabanlı politikalar tanımlanması.
- Oluşturulan politikaların düzenlenmesi.
    - Yeni sınıflar eklenmesi
    - Yeniden adlandırma.

  - Özellik tanımlarının düzenlenmesi.
  - Silme işlemi.
- Politikalar üzerinde sorguların gerçekleştirilmesi.

Bu hizmetler, sistem bileşenlerinin davranışlarını belirleyen politikaların, farklı alanların çeşitli uygulamaları tarafından kullanılmasını sağlamaktadır. Örgü sunuları içerisinde de erişim denetimi ve politika uygulamalarına uygun birçok durum oluşmaktadır. Ontoloji Tabanlı Erişim Denetimi kullanılarak kullanıcıya sunulabilecek sunuların bazıları aşağıda kısaca örneklendirilmektedir:

- ***E-turizm:*** Kullanıcı istekleklerine göre uyarlanmış turizm programlarına olan gereksinimin artması ile turizm ajansları müşterilerine kurallardan haberdar olmadan çeşitli seçenekleri sunabilecekleri teknolojilere gereksinim duymaktadırlar. Ontoloji Tabanlı Erişim Denetimi ile bu gereksinim karşılanmaktadır. Bu tez kapsamında verilen durum çalışması e-turizm alanı için geliştirilmiştir.
- ***E-sağlık:*** Günümüzün en popüler konularından biri olan e-sağlık uygulamalarında Anlamsal Web uygulamalarına duyulan gereksinim gözönüne alındığında başarılı bir e-sağlık ve Anlamsal Web sunuları bütünleşmesi için erişim denetiminin sağlık dizgelerine uygulanması gerekmektedir. Sağlık dizgelerinde kişilerin bilgi gizliliğinin sağlanması, hastalıkların tanısı ve tedavisi sırasında kullanılacak koşulların belirtilmesi gibi sunuların verilmesinde erişim denetiminin sağlanması ve

politikaların kullanılması gerekmektedir.

- ***Eğitim Dizgeleri:*** Anlamsal Web'in tanımladığı "anlamın tanımlanması (*expression of a meaning*)" eğitim sistemlerindeki birçok konuyla ilişkilendirilebilmektedir. Anlamsal Web tabanlı eğitim dizgeleri ile ilgili uygulamalarda politikalar kullanılarak eğitim dizgeleri yönetimi etkili bir şekilde gerçekleştirilebilir. Anlamsal Web teknolojileri kullanılarak; üniversitelerde kullanılan yönetmeliklerin tanımlanmasında, danışman-öğrenci arasındaki ilişkilerin belirtilmesinde, laboratuvar-yazıcı gibi yer ve donatımların kullanım kurallarında, düzenlenen konferansların düzenlenmesinde politikalar ve erişim denetim düzenekleri kullanılmaktadır.
- ***E-devlet:*** Günümüzde yaygınlaşan e-devlet uygulamalarının kapsamlı bir biçimde verilebilmesi için gizlilik ve erişim denetiminin sağlanması gerekmektedir. Kanunlar, vatandaşlık sunuları, yönetsel süreçler gibi e-devlet uygulamaları politikalar için ideal bir etki alanıdır.
- ***E-iş:*** Anlamsal Web verilerin ve veriler arasındaki ilişkilerin belirtilmesini sağlamaktadır. E-iş, iş süreçlerinin düzenleşiminin sağlanabilmesi için Anlamsal Web teknolojilerinin uygulanması ile bilginin yönetilmesidir. İş süreçlerinin örgü sunusu bağlamı olarak gerçekleştirilebilmesi için güvenlik önem kazanmaktadır (Huang, 2005). İş kuralları genellikle kısıtlar biçiminde gösterildiğinden iş kuralları politikaların kullanılması ile etkili bir biçimde gösterilmektedir (Huang, 2005). Farklı etki alanlarından ortaya çıkan güvenlik

gereksinimleri, örneğin; yasal problemler, gizlilik ve iş kuralları gibi, politikalar kullanılarak giderilmektedir (Huang, 2005).

- ***Mobil Uygulamalar:*** Cep telefonları, sayısal özel sekreterler (*Personal Digital Assistants, PDA*) ve kişisel bilgisayarlar mobil bilgi erişimini günlük yaşamın bir parçası durumuna getirmiştir. Anlamsal Web teknolojileri mobil uygulamalara çeşitli üstünlükler sağlamaktadır (Lassila, 2005). Bu, hareketlilik (mobilite) kavramınının beraberinde güvenlik ve gizlilik gibi kavramlarıda getirmektedir. Bilgiye ve sunuya erişimin kısıtlanması politikalar kullanılarak gerçekleştirilmelidir.

6. DURUM ÇALIŞMASI

Bu bölümde, Ontoloji Tabanlı Erişim Denetimi için kullanılan durum çalışması açıklanacaktır. Geliştirilen ontolojiler Protégé[1] ontoloji geliştirme aracı kullanılarak oluşturulmuştur.

4P Yazılım Şirketi yıllık şirket toplantısını Paris'te Hilton Paris otelinde gerçekleştirmektedir. Yazılım geliştiriciler, yöneticiler ve ortaklar bu toplantıya katılabilmektedir. Cem yazılım geliştirici, Can ise yazılım geliştirme takımının müdürüdür. Can Paris'e eşi Selin ile gitmeyi planlamaktadır. Ali ise 4P şirketinin ortak şirketlerinden birisini temsil etmektedir.

4P şirketinin otel ile yaptığı anlaşmaya göre şirket çalışanları otelin bütün olanaklarından yararlanabilmektedir. Şirket çalışanlarının otelin açık büfe ve buhar banyosu gibi fazladan hizmetleri için herhangi bir ödeme yapmalarına gerek yoktur. Ancak oda manzarası seçiminde otelin diğer konukları istedikleri oda manzarasını seçebilirken 4P şirketi çalışanları oda manzarası seçiminde Eiffel manzaralı odaları seçememektedir. Ayrıca, Hilton Paris otelinin bütün konukları otelin kablosuz internet erişiminden faydalanabilmektedir. Otelin bir diğer kuralı da oda-kahvaltı seçimi yapmış olan konukların açık büfeden yararlanamayacağıdır.

[1] Protégé, http://protege.stanford.edu/ .

6.1 Politika Ontolojisi Tanımlama Adımları

Politika ontolojisi tanımlanırken izlenen adımlar aşağıdaki yer almaktadır:

1. Rei politika ontolojilerinin isim uzayı (*namespace*) tanımlamaları yapılmaktadır.
2. `entity:Variable` sınıfı yaratılmaktadır. Kurallar oluşturulurken kullanılacak olan aktörler bu sınıfın altında yaratılan örneklerdir.
3. `constraint:SimpleConstraint` sınıfı yaratılmaktadır. Bu sınıf altında oluştulacak olan kısıt örnekleri yer almaktadır. Bu örnekler; `constraint:subject`, `constraint:predicate` ve `constraint:object` nesne özellikleri kullanılarak tanımlanmaktadır.
    a. Eğer karmaşık kısıt tanımlamaları yapılacaksa, yapılacak olan tanıma göre `constraint:And`, `constraint:Or` ve `constraint:Not` sınıfları yaratılmaktadır. `constraint:And` ve `constraint:Or` sınıflarının örnekleri `constraint:First` ve `constraint:Second` nesne özelliklerine iyedir. `constraint:Not` sınıfının örnekleri ise sadece `constraint:First` nesne özelliğine iyedir. Bu özellikler değerlerini `constraint:SimpleConstraint` sınıfının örneklerinden almaktadır.
4. `action:DomainAction` sınıfı yaratılmaktadır. Bu sınıfın

altında oluşturulan örnekler kurallar için kullanılacak olan eylemlerin tanımlamalarıdır. Bu tanımlamalar, eylemin aktörünü belirten `action:actor`, eylemin yerini gösteren `action:location` ve `constraint:SimpleConstraint` sınıfından ya da karmaşık kısıt değerlerinden birini alan `action:precondition` nesne özelliklerini kullanılmaktadırlar.

5. Oluşturulacak politika nesnesinin türüne göre `deontic:Permission`, `deontic:Prohibition`, `deontic:Obligation` ya da `deontic:Dispensation` sınıfları yaratılmaktadır. Bu sınıfların altında oluşturulan örnekler, ilgili eylemi belirten `deontic:action`, aktör değerini alan `deontic:actor` ve `constraint:SimpleConstraint` sınıfından ya da karmaşık kısıt değerlerinden birini alan `deontic:constraint` nesne özelliklerini kullanmaktadırlar.
6. Oluşturulan politika nesnelerinin onaylanması için `policy:Granting` sınıfı yaratılmaktadır. Bu sınıfın örnekleri, ilgili politika nesnesinin değerini alan `policy:deontic` ve politikanın aktör değerini alan `policy:to` nesne özelliklerini kullanmaktadırlar.
7. Politikanın oluşturulmasın son adımında, `policy:Policy` sınıfı yaratılmaktadır. Bu sınıfın altında yaratılan örnekler, politikanın aktörünü belirten `policy:actor`, ilgili eylemi belirten `policy:action` ve bir önceki adımda yapılan onaylamayı belirten `policy:grants` nesne özelliklerini kullanmaktadırlar.

6.2 Politika Ontolojileri

Oluşturulan politika ontolojileri üç biçimdedir. Bunlar:

- Etki Alanı Politika Ontolojisi
- Profil Tabanlı Politika Ontolojisi
- Seçenek Tabanlı Politika Ontolojisi

Bu bölümde sırası ile bu politika ontolojileri incelenecektir.

6.2.1 Etki alanı politika ontolojisi

Etki alanı politikalarında, politikalar ilgili etki alanı kurallarına göre oluşturulmaktadır. Durum çalışması kapsamında ele alınan turizm etki alanı ontolojisi temel alınarak `HotelPolicy` politikası oluşturulmuştur. Otel ile ilgili oluşturulan politikaların bir kısmı şu şekildedir:

Hilton Paris otelinde odalarda sigara içmeye izin verilmemektedir. (Yasak)

$$Accomodation(HiltonParis) \wedge hasRoom(HiltonParis, Guestroom) \wedge smokingAllowed(Guestroom, false) \rightarrow Prohibition(Smoking, HiltonParis)$$

Hilton Paris otelinde evcil hayvanlara izin verilmemektedir. (Yasak)

$$Accomodation(HiltonParis) \wedge petsAllowed(HiltonParis, false) \rightarrow Prohibition(OwningPet, HiltonParis)$$

Hilton Paris otelinin bütün ziyaretçileri kablosuz internet bağlantısından yararlanabilir. (İzin)

$$Accomodation(HiltonParis) \wedge wirelessConnection(HiltonParis, true) \wedge wirelessConnection(Visitors, true) \rightarrow Permission(InternetAccess, HiltonParis)$$

Turizm etki alanı ontolojisi olarak DERI tarafından geliştirilen e-tourism[1] ontolojisi temel alınmıştır. Bu ontolojinin sınıfları aşağıdaki gibidir:

- Konaklama (Accomodation)
- Etkinlik (Activity)
- İletişim Verisi (Contact Data)
- Gün Saat (Date Time)
    - Açılış Saatleri (Opening Hours)
    - Dönem (Period)
        - Gün (DatePeriod)
        - Saat (TimePeriod)
    - Mevsim (Season)

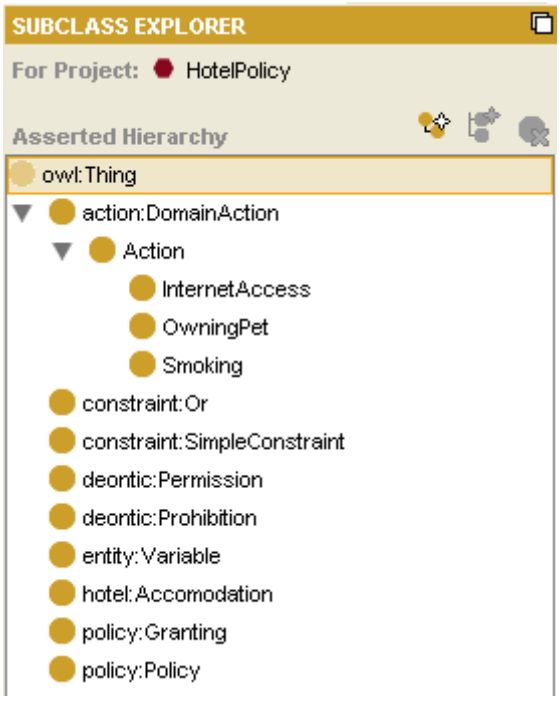
[1] http://e-tourism.deri.at/ont/e-tourism.owl

- Olay (Event)
- Altyapı (Infrastructure)
- Dil (Language)
- Yer (Location)
    - GPS Koordinatları (GPSCoordinates)
    - Posta Adresi (PostalAddress)
- Oda (Room)
    - Konferans Odası (ConferenceRoom)
    - Misafir Odası (Guestroom)
- Bilet (Ticket)

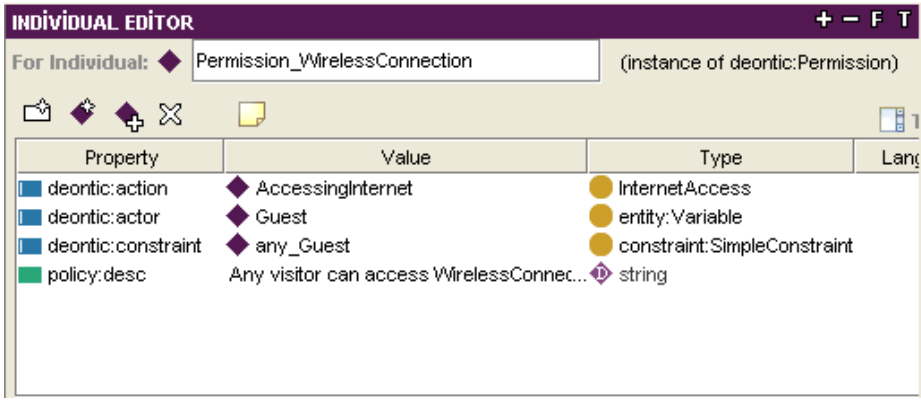
Durum çalışması kapsamında, bu etki alanı ontolojisinin özelliklerine çeşitli eklemeler yapılarak ontoloji genişletilmiştir. Genişletilmiş turizm ontolojisi `http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/I_Tour_mdfy.owl` adresinde yer almaktadır.

Turizm etki alanı ontolojisi kavramları temel alınarak otel politika ontolojisi oluşturulmuştur. Otel politika ontolojisinde ”*Hilton Paris otelinin bütün ziyaretçileri kablosuz internet bağlantısı hizmetinden yararlanabilirler.*” izini tanımlanmıştır. Politika ontolojisinde yer alan sınıflar Şekil 6.1’de görülmektedir. Erişim izini `deontic:actor`, `deontic:action` ve `deontic:constraint` kavramlarından oluşmaktadır. Bu kavramlar, Rei politika dilinin ReiDeontic ontolojisi altında yer almaktadır. `deontic:actor` eylem ile ilgili kişiyi, `deontic:action` eylemi ve `deontic:constraint`’de eylem ile ilgili

kısıtı tanımlamaktadır. Şekil 6.2'de erişim izini ve kavramları yer almaktadır.

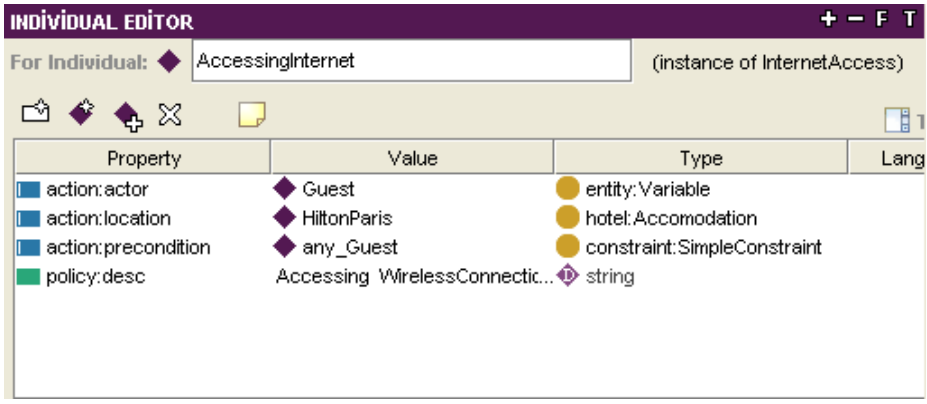


Şekil 6.1 Otel politika ontolojisi sınıfları.



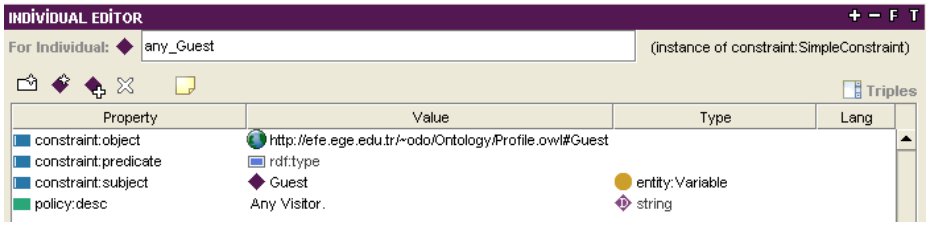
Şekil 6.2 Kablosuz internete erişim izni.

Erişim izninin eylemini belirten ”`AccessingInternet`” değeri `deontic:action` kavramı ile ilişkilendirilmektedir. ”`AccessingInternet`” değerinin tanımlaması Şekil 6.3’de görülmektedir.



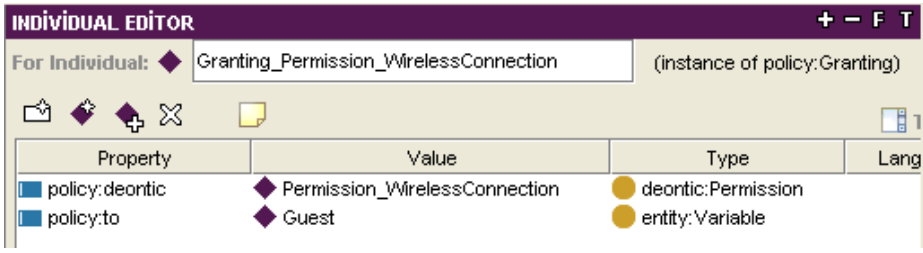
Şekil 6.3 İnternete erişim eyleminin tanımlanması.

Erişim izninin uygulanacağı kısıtlamayı belirten `deontic:constraint` kavramının ”`any_Guest`” değeri ise Şekil 6.4’te yer almaktadır. Burada `constraint:object` özelliği değer olarak profil ontolojisinden gelen ”`http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/Profile.owl#Guest`” değerini almaktadır.



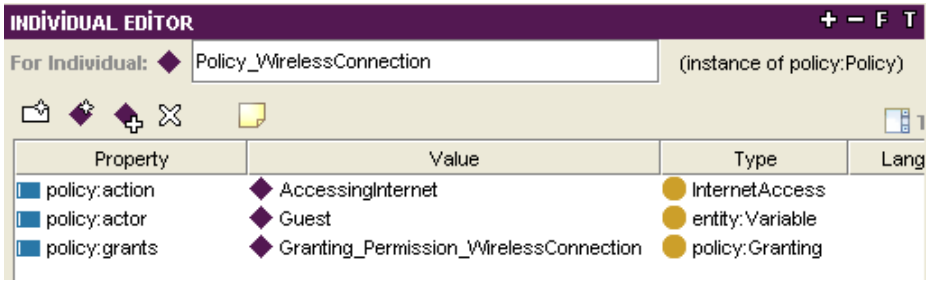
Şekil 6.4 Ziyaretçi kısıtının tanımlanması.

Erişim izininin tanımlanması için bu izinin onaylanması Şekil 6.5’te görülmektedir.



Şekil 6.5 Erişim izininin onaylanması.

Şekil 6.6'da yer alan politika örneği ile erişim izini tanımlanması son bulmaktadır. Bu politika örneği, `policy:action`, `policy:actor` ve `policy:grants` değer tanımlamalarının yapıldığı `policy:Policy` sınıfında yer almaktadır.

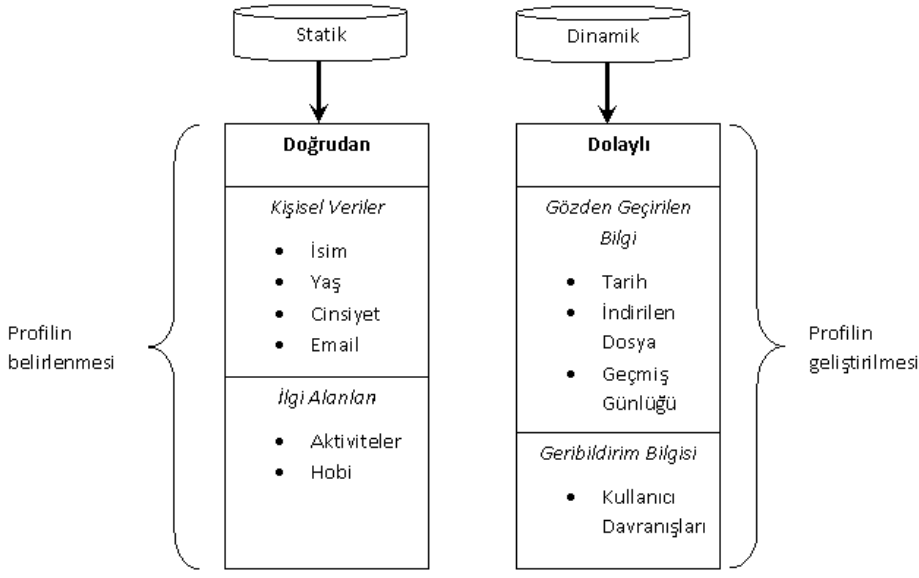


Şekil 6.6 Erişim izini politikasının tanımlanması.

6.2.2 Profil tabanlı politika ontolojisi

Ontoloji Tabanlı Erişim Denetimi'nde profil tabanlı bir yaklaşım

kullanılmaktadır. Kullanıcı profillerinin modellenmesi kişiselleştirmenin önemli bir unsurunu oluşturmaktadır. Kullanıcı bilgilerinin toplanması ve bu bilgilerden kullanıcı profillerinin oluşturulabilmesi için çeşitli yöntemler bulunmaktadır. Kullanıcı bilgileri doğrudan (*explicitly*) ya da dolaylı (*implicitly*) olarak sağlanabilir (Jrad and Aufaure, 2007). Çevrimiçi kayıt formları, anketler ve eleştiriler doğrudan sağlanan bilgidir. Bu durağan profil olarak adlandırılmaktadır. Devingen profil olarak adlandırılan dolaylı profillemede ise çerezler (*cookies*) ve örgü sunucu günlük kütükleri aracılığı ile her bir kullanıcının seçenekleri ve davranış şekli kaydedilip çözümlenebilmektedir. Şekil 6.7'de durağan ve devingen profil yapısı görülmektedir (Jrad and Aufaure, 2007).



Şekil 6.7 Durağan ve devingen profil yapısı.

Profil tabanlı politika yönetimi bu tez çalışmasında kullanılan yaklaşımlardan bir tanesidir. Politika, kullanıcı profil bilgisini kullanarak kullanıcının ideal davranışlarını belirlemektedir. Profilleme aynı sınıftaki kullanıcıların kümesinin oluşturulmasıdır. Profiller bilginin belirli bir kullanıcıya uyarlanmasına yardımcı olmaktadır (Dzbor and Motta, 2008). Kullanıcılar yaş ve meslek gibi ortak özellikler ya da kullanıcıların yapmış oldukları ortak ilgileri dikkate alınarak gruplandırılmaktadır. Örneğin, yaşı 18 ve 35 arasında olan kullanıcılar "*Genç*" profiline, Profesör olarak çalışan kullanıcılar "*Akademik*" profiline, diyabetik olan kullanıcılar ise "*Diyabetik*" profilinde olmaktadırlar. Roller kullanıcılar olmadan da yaratılabilirken, profiller kullanıcılar olmadan yaratılamamaktadırlar. Profillemede, kullanıcı olmadan profil olmamaktadır. Bu nedenle profillemede kullanıcı bilgisine gereksinim duyulmaktadır. Ancak, rollerin kullanılması durumunda herhangi bir kullanıcı bilgisine gereksinim duyulmamaktadır. RBAC modelinde yer alan sınırlamalar nedeni ile bu tez çalışmasında roller yerine profiller kullanılmaktadır. Profil tabanlı politikalar oluşturulurken üst profil ontolojisi kullanılarak yaratılmış olan profil ontolojisi temel alınmaktadır.

Profil ontolojisi olan http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/Profile.owl[1]'da profil, profilin adı, yaş ve meslek bilgileri tanımlanmaktadır. Profil ontolojisinin kullandığı bu değerler ile

[1] Okan Bursa (okan.bursa@ege.edu.tr) tarafından geliştirilmiştir.

ilgili sınıflar ve özellikler FOAF[1] (Friend Of A Friend) ve üst profil (`MetaProfile.owl`)[1] ontolojilerinden alınmaktadır. Üst Profil ontolojisi profil tanımlamak için gerekli üst verileri (yaş, gelir, profil adı vb.) tanımlamaktadır. Örneğin akademisyen profili için Profile.owl ontolojisinde tanımlanan `Academician` örneği 3 özelliğe sahiptir. Bunlar; akademisyenin yaşı (`hasAge`), mesleği (`hasOccupation`) ve adıdır (`hasName`). Akademisyen profilinin yaş örneğinin değeri üst profil ontolojisinde belirtilmiş olan en büyük (`AcademicAgeValueMax`) ve en küçük yaş (`AcademicAgeValueMin`) özellikleri kullanılarak tanımlanmaktadır. Bu değerlerin tanımlanmasında ise FOAF ontolojisinin `ageValue` veri türü özelliği kullanılmaktadır. Akademisyenin meslek örneği tanımı yapılırken FOAF'ın `can-be` veri türü özelliği, akademisyen adı örneği tanımlamasında ise üst profil ontolojisinin `name` veri türü özelliği kullanılmaktadır.

Durum çalışması kapsamında geliştirilen bazı profil kuralları aşağıdaki gibidir:

Bir turist profili ekstra bir ödeme yaparsa buhar banyosundan yararlanabilir.(Zorunluluk)

$$Profile(Tourist) \wedge hasPayment(extraBill, true) \rightarrow Obligation(Use, SteamBath)$$

[1] FOAF, http://www.foaf-project.org/ .

Turist profili Eiffel manzaralı odada kalabilir. (İzin)

$$Profile(Tourist) \wedge hasView(Guestroom, Eiffel) \rightarrow Permission(StayIn, Guestroom)$$

Şirket Çalışanı profili Eiffel manzaralı odalarda kalamaz. (Yasak)

$$Profile(CompanyEmployee) \wedge hasView(Guestroom, Eiffel) \rightarrow Prohibition(StayIn, Guestroom)$$

Kullanıcı Şirket Çalışanı profilinde ise buhar banyosu için ekstra ödeme yapmasına gerek yoktur. (Özel İzin)

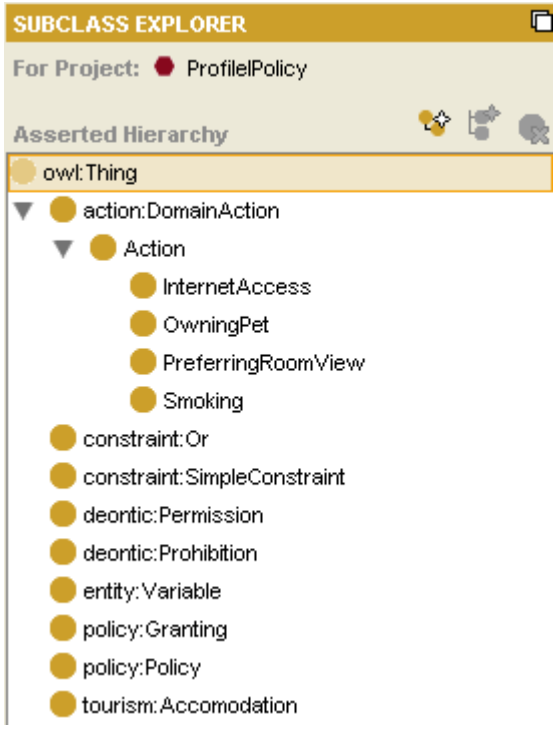
$$Profile(CompanyEmployee) \wedge hasPayment(extraBill, false) \rightarrow Dispensation(Use, SteamBath)$$

Profil tabanlı politika ontolojisinde eylemlerin aktörleri olarak politika ontolojisinin ”`entity:Variable`” sınıfı altında yaratılan değerler yerine profil ontolojisinde yer alan değerler kullanılmaktadır. Profil ontolojisi olarak `http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/Profile.owl` kullanılmaktadır. Burada `entity:Variable` sınıfı ile profil ontolojisi arasında herhangi bir eşleme bulunmamaktadır. `entity:Variable` sınıfı içerisinde politikada yer alan varlık örnekleri tutulmaktadır. Profil tabanlı politikalar tanımlanırken, bu varlıklar `entity:Variable` sınıfı yerine anlamsal olarak daha zengin olan profil ontolojisinden alınmaktadır.

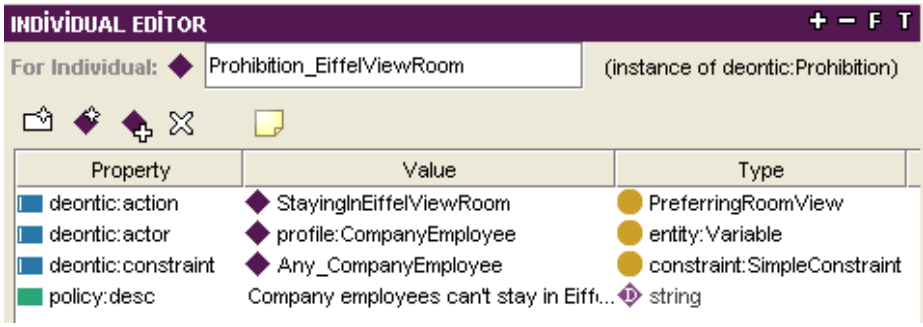
Durum çalışması kapsamında oluşturulan profil tabanlı politika ontolojisinde ”*Şirket çalışanı profilinde olan kişiler Eiffel manzaralı odalarda kalamazlar.*” yasağı tanımlanmıştır. Politika ontolojisinde yer

alan sınıflar Şekil 6.8'de görülmektedir. Tanımlanan yasak politika nesnesi `deontic:actor`, `deontic:action` ve `deontic:constraint` kavramlarından oluşmaktadır.

Şekil 6.9'da bu yasak ile ilgili kavramlar yer almaktadır. Otel politika ontolojisinden farklı olarak burada `deontic:actor` kavramı `entity:Variable` sınıfındaki değeri değil profil ontolojisinde tanımlanmış olan `profile:CompanyEmployee` değerini almaktadır.

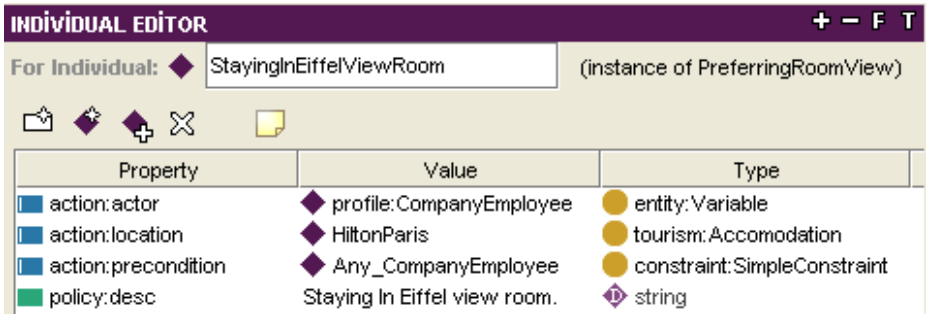


Şekil 6.8 Profil tabanlı politika ontolojisi sınıfları.



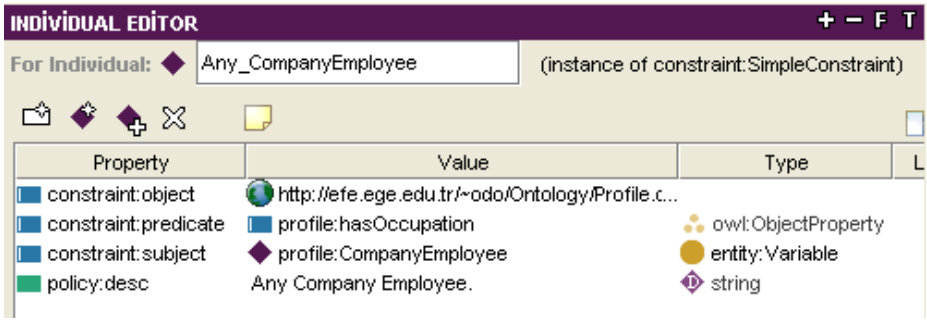
Şekil 6.9 Şirket çalışanları için Eiffel manzaralı oda yasağı tanımlanması.

Yasak ile ilgili eylemi belirten `StayinInEiffelViewRoom` değeri `deontic:action` kavramı ile ilişkilendirilmektedir. Bu eylem `Action` sınıfının alt sınıfı olan `PreferringRoomView` sınıfının bir değeridir. `StayingInEiffelViewRoom` değerinin tanımlaması Şekil 6.10'da görülmektedir.



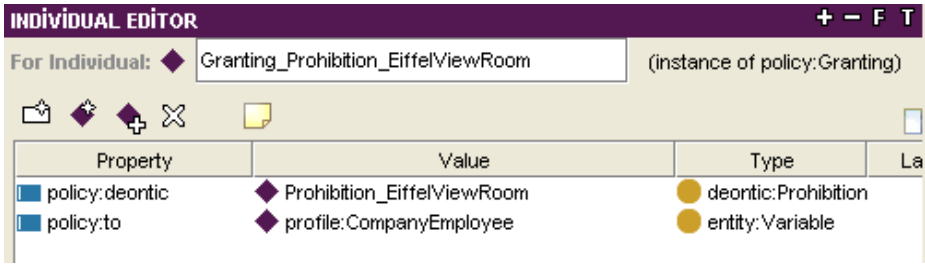
Şekil 6.10 Eiffel manzaralı odada kalma eyleminin tanımlanması.

Yasağın uygulanacağı kısıtlamayı belirten `deontic:constraint` kavramının `any_CompanyEmployee` değerinin tanımlanması Şekil 6.11'de yer almaktadır. Burada `constraint:object` özelliği profil ontolojisinden gelen `http:// efe.ege.edu.tr/~ozgucan/dr/Ontoloji/Profile.owl#CompanyEmployeeOccupation` değerini, `constraint: predicate` özelliği değer olarak profil ontolojisinden gelen `profile:hasOccupation` özelliğini, `constraint:subject` özelliği ise `profile:CompanyEmployee` değerini almaktadır.

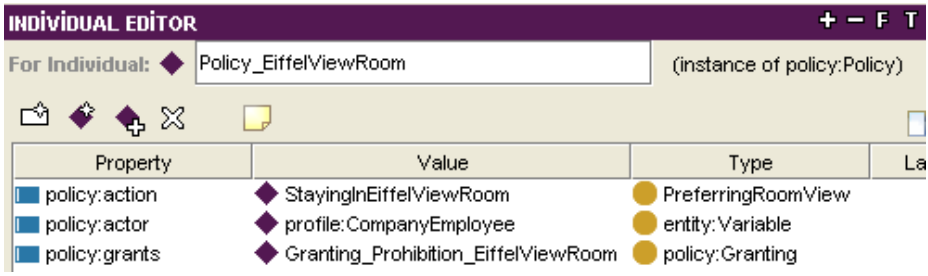


Şekil 6.11 Şirket çalışanı kısıtının tanımlanması.

Yasağın tanımlanması bu yasağın onaylanması ve daha sonra da politika olarak belirtilmesi ile son bulmaktadır. Yasağın onaylanması Şekil 6.12'de, politika olarak belirtilmesi ise Şekil 6.13'de görülmektedir.



Şekil 6.12 Yasağın onaylanması.



Şekil 6.13 Yasak politikasının tanımlanması.

6.2.3 Seçenek tabanlı politika ontolojisi

Seçenek, bir varlığın, özelliğin ya da kısıtın belirli bir kümedeki başka bir varlık, özellik ya da kısıta olan üstünlüğünü belirten bir durumdur (Tapucu vd., 2008). Politikalar bu seçeneklerin yönetilmesi için kullanılmaktadır. OBAC modelinde, seçenek tabanlı poltikaların temel aldığı seçenekler, seçenek üst ontolojisi kullanılarak oluşturulmaktadır. Seçenek politikası, ilgili etki alanında aynı seçenekleri olan kullanıcılar için kurallar tanımlamaktadır. Örneğin;

Oda-kahvaltı seçimi yapan kişiler açık büfeden yararlanamaz. (Yasak)

$$Profile(Guest) \wedge hasPreference(bedBreakfast, true) \rightarrow Prohibition(Use, SnackBar)$$

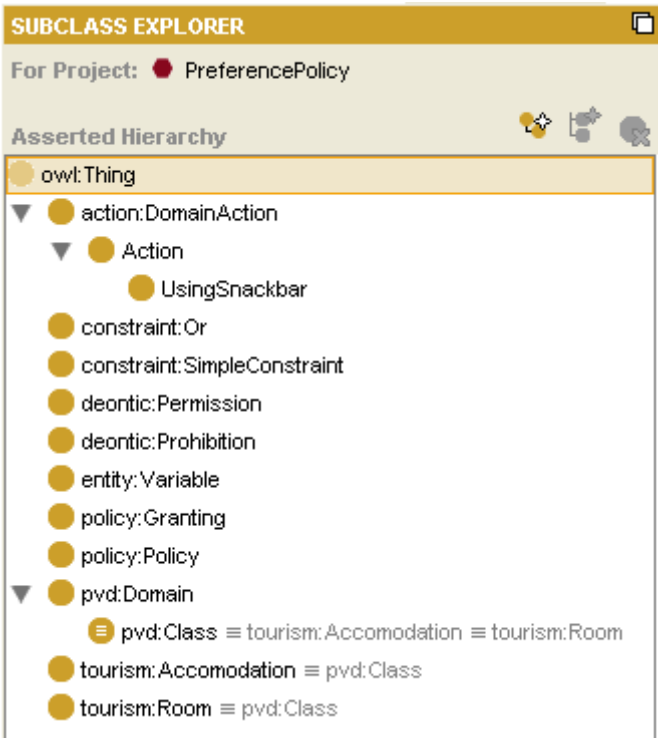
Seçenek tabanlı politika ontolojisinde eylemlerin kısıtlamaları belirtilirken üst seçenek ontolojilerinde tanımlanmış olan değerler kullanılmaktadır. Üst seçenek ontolojileri olarak Üst Seçenek Ontolojisi[1] (Preference Meta Ontology - PMO), Seçenek Görüntüleme Ontolojisi[1] (Preference View Domain - PVD) ve Etki Alanı Seçenek Ontolojisi[1] (PVD_Domain) ontolojileri kullanılmaktadır. Bu ontolojiler sırası ile http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/PMO.owl, http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/ PVD.owl ve http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/PVD_Domain.owl adreslerinde bulunmaktadır.

PMO (Preference Meta Ontology), kullanıcı seçeneklerinin tanımlanması için gerekli olan varlıkları tanımlamaktadır. Kullanıcının kişisel seçeneklerini gösterebilmek için farklı seçenek türleri tanımlanmaktadır (Tapucu vd., 2008). PVD (Preference View Domain), etki alanı ontolojisini sınıflar, özellikler ve örnekler olarak ayrıştırmakta ve ontoloji kaynaklarını sıradüzensel bir şekilde tutmaktadır (Tapucu vd., 2008). PVD_Domain ontolojisi ise PVD ile ayrıştırılan etki alanı ontolojisinin örneklerini tutmaktadır.

Durum çalışması kapsamında oluşturulan seçenek tabanlı politika

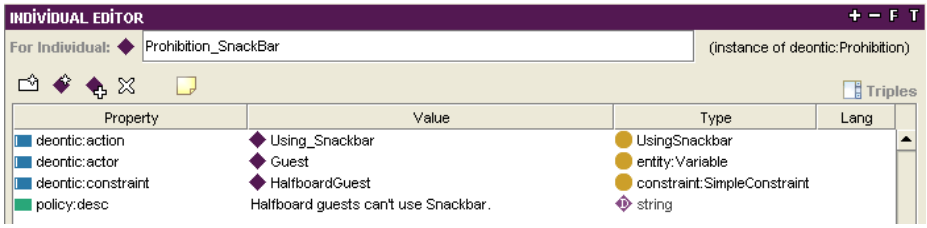
[1] Dilek Tapucu (dilektapucu@iyte.edu.tr) tarafından geliştirilmiştir.

ontolojisinde ”*Oda-kahvaltı seçimi yapmış ziyaretçiler açık büfeden yararlanamazlar.*” yasağı tanımlanmıştır. Politika ontolojisinde yer alan sınıflar Şekil 6.14’de görülmektedir.



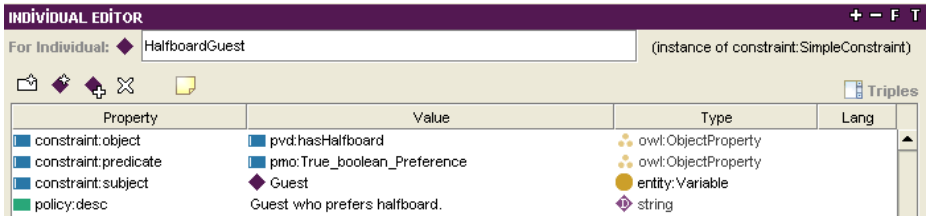
Şekil 6.14 Seçenek tabanlı politika ontolojisi sınıfları.

Tanımlanan yasak politika nesnesi `deontic:actor`, `deontic:action` ve `deontic:constraint` kavramlarından oluşmaktadır. Şekil 6.15’de bu yasak ile ilgili kavramlar yer almaktadır.



Şekil 6.15 Oda-Kahvaltı seçimi yapanlar için açık büfe yasağı tanımlanması.

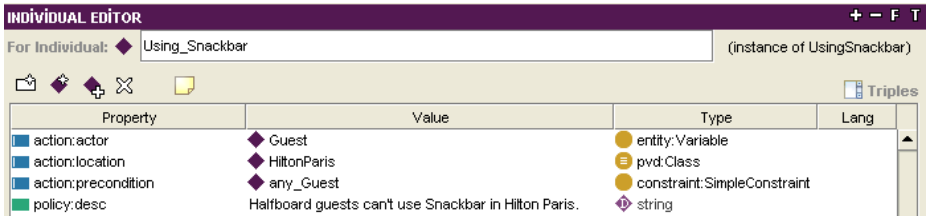
Yasağın uygulanacağı kısıtlamayı belirten `deontic:constraint` kavramının `HalfboardGuest` değerinin tanımlanması Şekil 6.16'da yer almaktadır. Burada `constraint:object` özelliği PVD seçenek üst ontolojisinden gelen `pvd:hasHalfboard` değerini, `constraint:predicate` özelliği ise PMO seçenek üst ontolojisinden gelen `pmo:True_boolean_Preference` değerini almaktadır.



Şekil 6.16 Oda-Kahvaltı seçimi yapan ziyaretçi kısıtının tanımlanması.

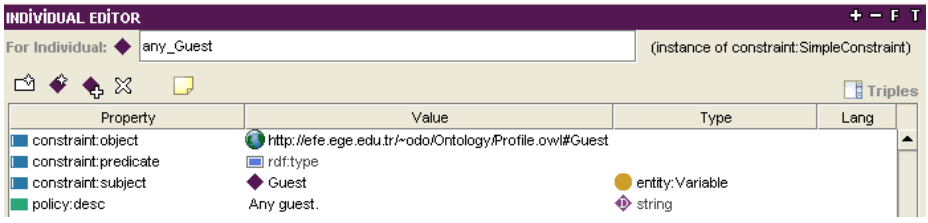
Yasak ile ilgili eylemi belirten `Using_Snackbar` değeri `deontic:action` kavramı ile ilişkilendirilmektedir. Bu eylem `Action`

sınıfının alt sınıfı olan `UsingSnackbar` sınıfının bir değeridir. `Using_Snackbar` değerinin tanımlaması Şekil 6.17'de görülmektedir.



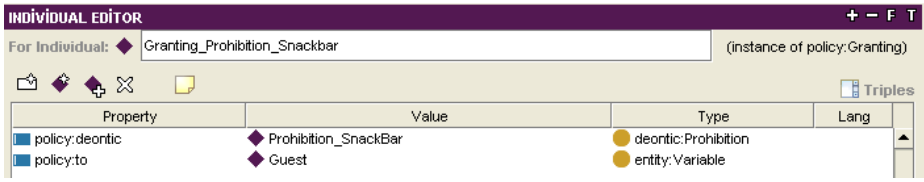
Şekil 6.17 Açık büfeyi kullanma eyleminin tanımlanması.

`Using_Snackbar` eyleminde yer alan `any_Guest` kısıtlamasının tanımlanması Şekil 6.18'de yer almaktadır. Burada `constraint:object` özelliği profil ontolojisinden gelen `http://efe.ege.edu.tr/~ozgucan/dr/Ontoloji/Profile.owl#Guest` değerini almaktadır.

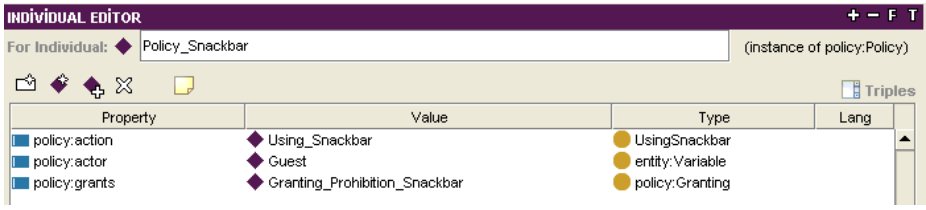


Şekil 6.18 Konuk kısıtının tanımlanması.

Şekil 6.19 ve Şekil 6.20 sırası ile yasağın onaylanmasını ve politika olarak belirtilmesini göstermektedir.



Şekil 6.19 Yasağın onaylanması.



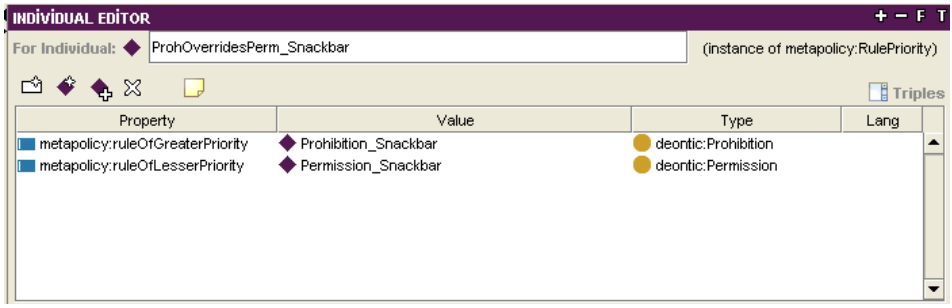
Şekil 6.20 Yasak politikasının tanımlanması.

6.3 Politika Çelişkilerinin Çözümü Ontolojisi

Politikalar oluşturulurken ortaya çıkabilecek çelişkilerin çözümü kural üstünlükleri ve politika üstünlüklerinin tanımlanması olmak üzere iki şekilde gerçekleştirilmektedir.

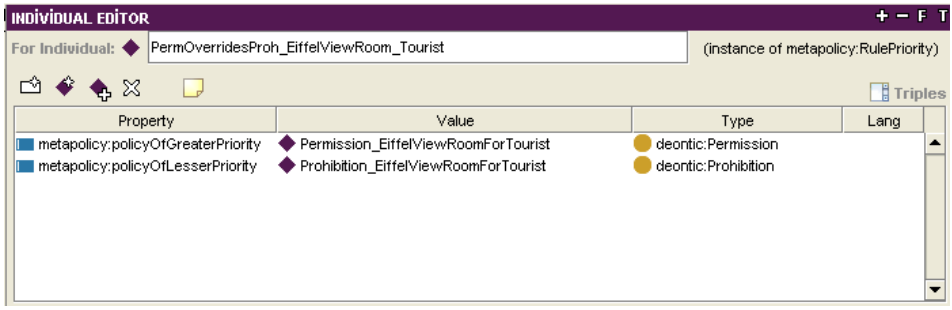
Kural üstünlüklerinin tanımlanmasında Rei politika dilinin ReiMetaPolicy ontolojisinde yer alan `metapolicy:ruleOf`

`GreaterPriority` ve `metapolicy:ruleOfLesserPriority` nesne özellikleri kullanılmaktadır. Şekil 6.21'de bir kural çelişkisinin çözümü görülmektedir. Burada açık büfe kullanımı ile ilgili yasak kuralının izin kuralına üstünlüğü tanımlanmaktadır.



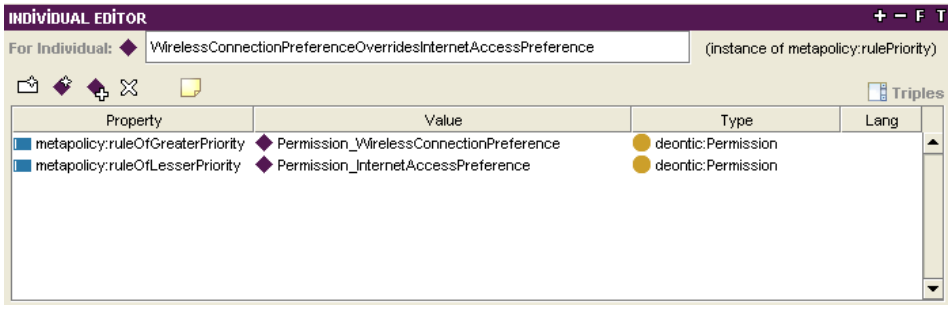
Şekil 6.21 Yasak kuralının izin kuralına üstünlüğünün tanımlanması.

Politika üstünlüklerinin tanımlanmasında `metapolicy:policy OfGreaterPriority` ve `metapolicy:policyOfLesser Priority` nesne özellikleri kullanılmaktadır. Şekil 6.22'de bir politika çelişkisinin çözümü yer almaktadır. Burada Eiffel manzaralı odalarda turistlerin kalması ile ilgili izin politkasının yasak politikasına üstünlüğü tanımlanmaktadır.



Şekil 6.22 İzin politikasının yasak politikasına üstünlüğünün tanımlanması.

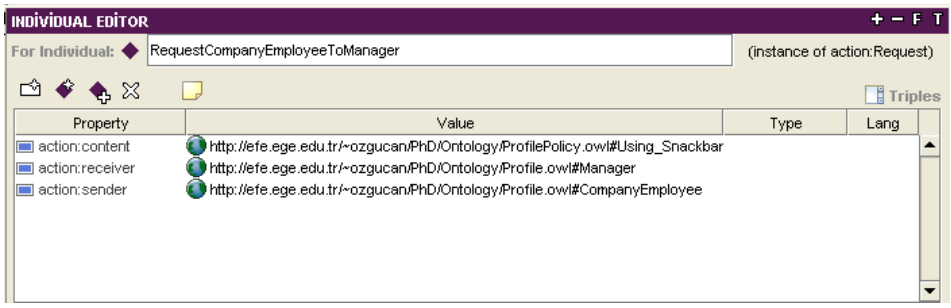
Kullanıcı seçimleri ile ilgili çelişki olduğunda yine kuralların çelişmesi ile çözümlenmektedir. Örneğin; `wirelessConnection` ve `internetAccess` seçimleri yapmış bir kullanıcı için `wirelessConnection` seçiminin `internetAccess` seçimine üstünlüğü Şekil 6.23'de tanımlanmaktadır.



Şekil 6.23 Seçenek çelişkisinin çözümü.

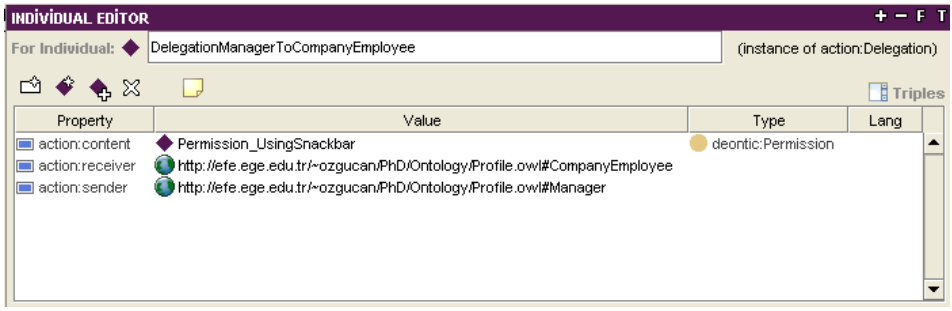
6.4 Konuşma Edimleri Ontolojisi

Konuşma edimleri politikaların daha az ayrıntılı olmasını ve merkezi olmayan güvenlik denetimini sağlamaktadır. Konuşma edimi ontolojisi oluşturulurken Rei politika dilinin ReiAction ve ReiDeontic ontolojileri kullanılmakta ve *istek*, *yetki aktarması*, *iptal* ve *yetkinin geri alımı* olmak üzere 4 konuşma edimi tanımlanmaktadır. Bu tezde kullanılan durum çalışmasına göre oluşturulan konuşma edimlerinde öncelikle şirket çalışanı şirket yöneticisinden açık büfeyi kullanma izni istemektedir. Bu amaçla oluşturulan `action:Request` sınıfında yer alan örneğin özellikleri ve değerleri Şekil 6.24'de görülmektedir.



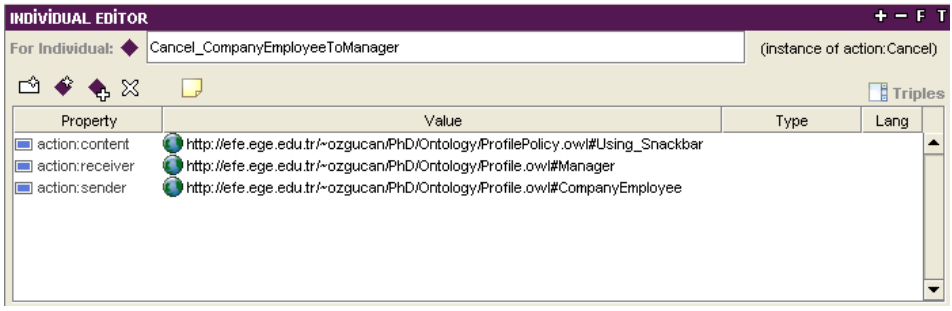
Şekil 6.24 İstek konuşma edimi.

Şekil 6.25'de şirket yöneticisinin çalışanına açık büfeyi kullanma yetkisini aktarması yer almaktadır. Bunun için `action:Delegation` sınıfı ve bu sınıfa ilişkin örnek oluşturulmaktadır.



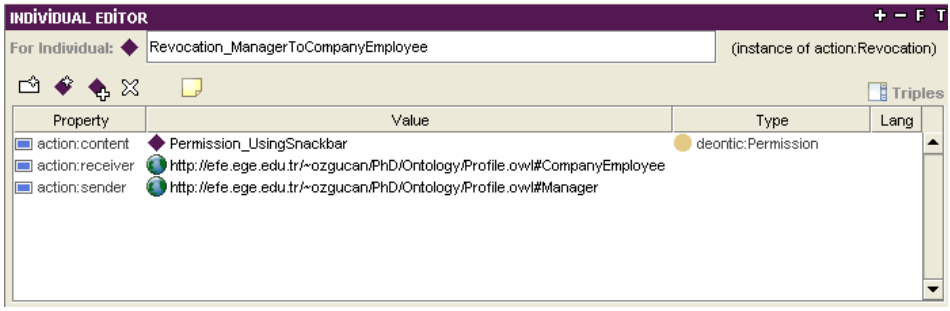
Şekil 6.25 Yetki aktarımı konuşma edimi.

Şirket çalışanının yöneticisine gönderdiği isteği iptal etmesi için `action:Cancel` sınıfı oluşturulmaktadır. Bu sınıfa ilişkin örneğin özellikleri ve değerleri Şekil 6.26'da görülmektedir.



Şekil 6.26 İptal konuşma edimi.

Şirket yöneticisinin çalışanına verdiği yetkiyi geri alması Şekil 6.27'de yer almaktadır. Yetkinin geri alınmasında `action:Revocation` sınıfı yaratılmaktadır.



Şekil 6.27 Yetkinin geri alımı konuşma edimi.

7. UYGULAMA GERÇEKLEŞTİRİMİ

Bir kullanıcı arayüzü aracılığı ile politikaların yaratılması, düzenlenmesi, silinmesi ve politikalar üzerinde sorguların yapılabilmesi için Ontoloji Tabanlı Erişim Denetimi uygulaması geliştirilmiştir. Bu uygulama Java programlama dili kullanılarak gerçekleştirilmiştir. Bu amaçla, Anlamsal Web uygulamalarının geliştiriminde sıklıkla kullanılan Jena Anlamsal Web Çatısından yararlanılmıştır.

7.1 Java

Java programlama dilinin kullanıldığı Ontoloji Tabanlı Erişim Denetimi uygulaması Eclipse[1] ortamı kullanılarak geliştirilmiştir. Arayüz tasarımında Eclipse ortamına uyumlu bir ek olan Jigloo[2] arayüz yapıcısı tercih edilmiştir.

7.2 Jena

HP Laboratuvarları tarafından geliştirilmiş olan Jena[3], Anlamsal Web uygulamalarının geliştirildiği bir Java çatısıdır. Anlamsal Web bilgi modeli ve dillerini kullanan uygulamaların kolay bir şekilde geliştirilmesini sağlamaktadır. RDF, RDFS, OWL ve SPARQL için

[1] Eclipse, http://www.eclipse.org .

[2] Jigloo, http://www.cloudgarden.com/jigloo .

[3] Jena, http://jena.sourceforge.net .

programlama ortamını sağlamakta ve kural tabanlı bir çıkarsama motorunu içermektedir. http://jena.sourceforge.net/downloads.html adresinden indirilebilen Jena açık kaynak kodu Java kütüphanesine aktarılarak kullanılmaktadır.

7.2.1 Jena Örnekleri

Bu bölümde Anlamsal Web uygulamalarının geliştiriminde kullanılan temel Jena işlemleri örneklendirilmektedir.

İsim uzaylarının yaratılması:

```
String ReiAction="http://www.cs.umbc.edu/~lkagal1/rei/
ontologies/ReiAction.owl#";
PrefixMapping ReiActionNS=tourismModelPolicy.setNsPrefix
("action", ReiAction);
```

Bir ontoloji modelinin yaratılması:

```
OntModel tourismModelPolicy = ModelFactory.createOntology
Model();
```

Boş bir OWL modelinin yaratılması:

```
OntModel tourismModelPolicy = ModelFactory.createOntology
Model(OntModelSpec.OWL_MEM, null);
Ontology tourismModelPolicyOntology=tourismModelPolicy.
createOntology(OntologyURI);
```

Bir nesne özelliğinin yaratılması:

```
ObjectProperty actionActor = tourismModelPolicy.create
ObjectProperty(ReiAction + "actor");
```

Bir veri türü özelliğinin yaratılması:

```
DatatypeProperty policyDesc = tourismModelPolicy.create
DatatypeProperty(ReiPolicy + "desc");
```

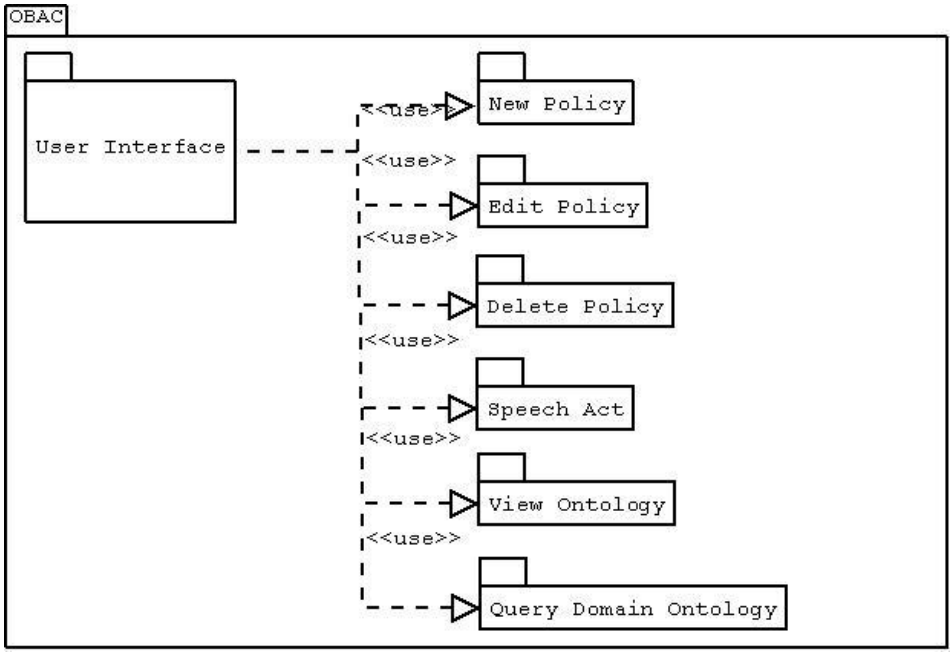
7.3 Politika Motoru Yapısı

Ontoloji Tabanlı Erişim Denetimi için geliştirilen politika motorunun yapısal kullanım görünümü Şekil 7.1'de yer aldığı gibi 6 birimden oluşmaktadır. Bunlar:

- Yeni Politika (New Policy)
- Politikanın Düzenlenmesi (Edit Policy)
- Politikanın Silinmesi (Delete Policy)
- Konuşma Edimi (Speech Act)
- Ontolojinin Görüntülenmesi (View Ontology)
- Etki Alanı Ontolojisinin Sorgulanması (Query Domain Ontology)

Şekil 7.1 ve bu alt bölümde yer alan şekiller UML[1] (Unified Modelling Language) kullanılarak geliştirilmiştir.

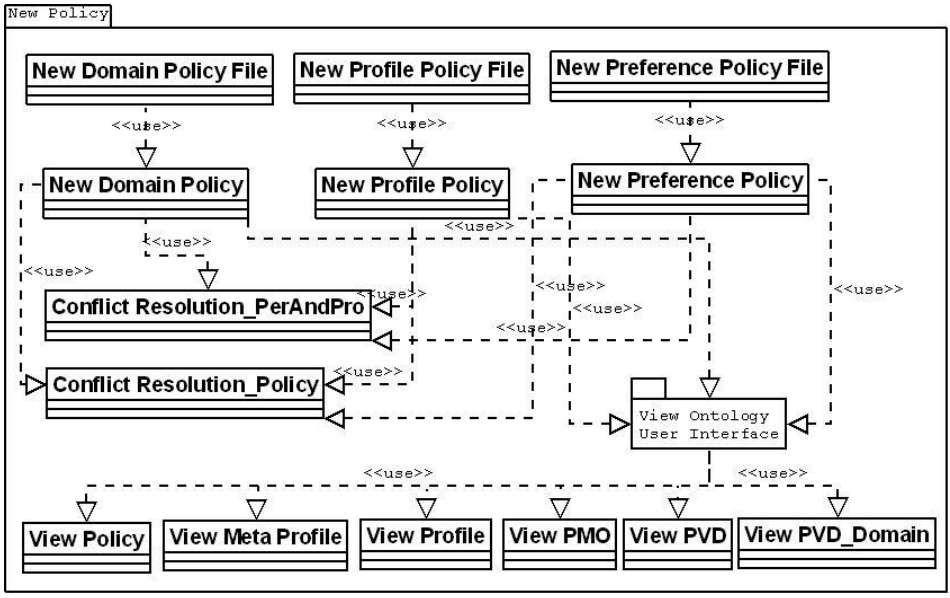
[1] UML, http://www.omg.org/technology/documents/formal/uml.htm .



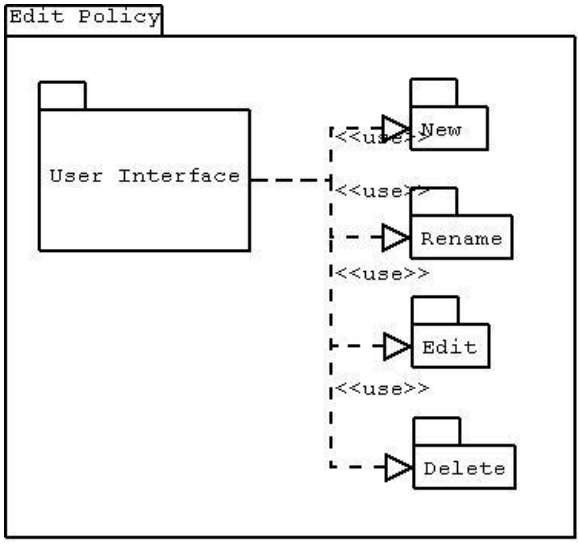
Şekil 7.1 Ontoloji tabanlı erişim denetimi politika motorunun yapısal kullanım görünümü.

Şekil 7.2'de yer alan yeni politika tanımının yapıldığı birimde, her bir yeni etki alanı, profil ve seçenek politika dosyası tanımının yapıldığı sınıflar; yeni etki alanı, profil ve seçenek sınıflarını kullanmaktadır. Bu sınıflarda izin ve yasak deontik politika kuralı çelişkisi çözümü, politika çelişki çözümü ve ontoloji görüntüleme sınıflarını kullanmaktadır.

Politika düzenleme birimi Şekil 7.3'de gösterilmiştir. Bu birim; *yeni* (new), *yeniden adlandırma* (rename), *düzenleme* (edit) ve *silme* (delete) birimlerinden oluşmaktadır.

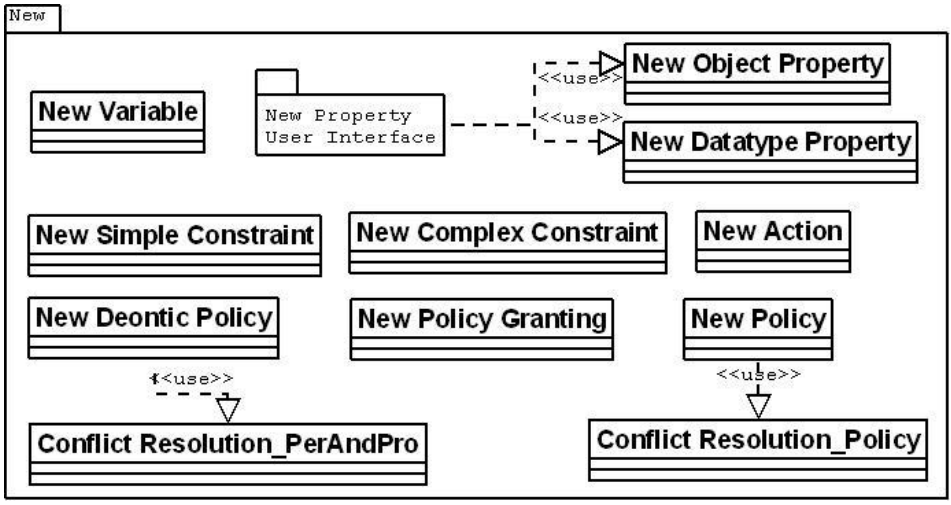


Şekil 7.2 Yeni politika tanımı birimi.



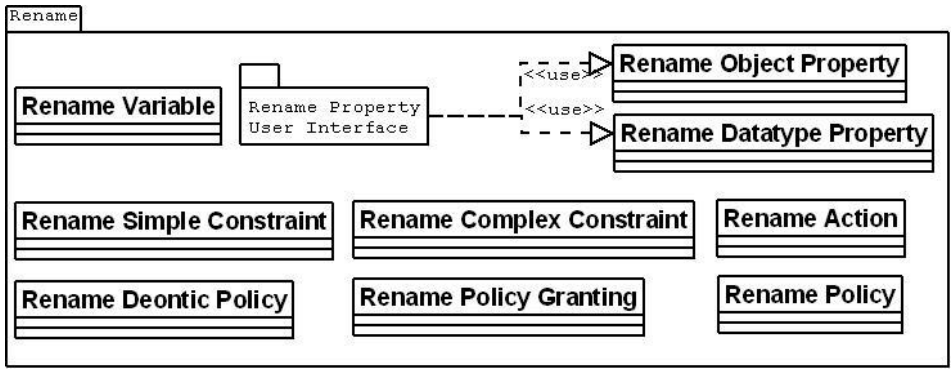
Şekil 7.3 Politika düzenleme birimi.

Politika düzenleme biriminin *yeni* birimi 9 sınıftan oluşmaktadır. *Yeni* biriminin yapısı Şekil 7.4'de yer almaktadır. Ayrıca, bu 9 sınıfa ek olarak, bu birimde yer alan `New Deontic Policy` ve `New Policy` sınıfları oluşabilecek çelişkilerin tespiti ve çözümü için sırası ile `Conflict Resolution_PerAndPro` ve `Conflict Resolution_Policy` sınıflarını kullanmaktadır.

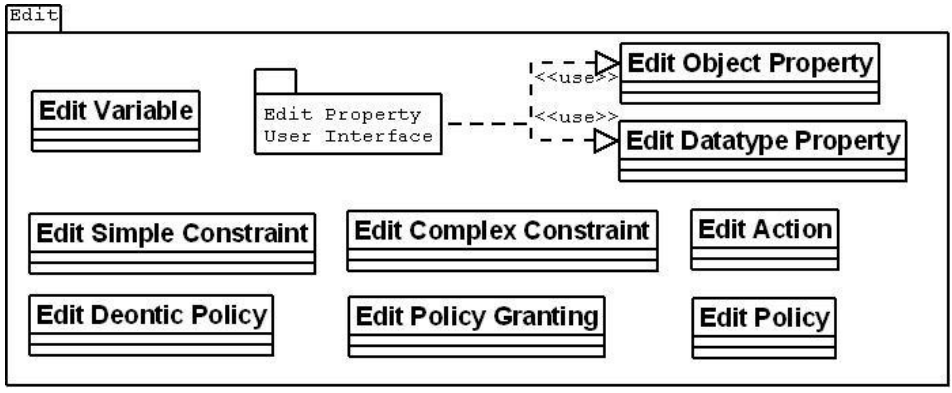


Şekil 7.4 Politika düzenlemesi biriminin yeni birimi.

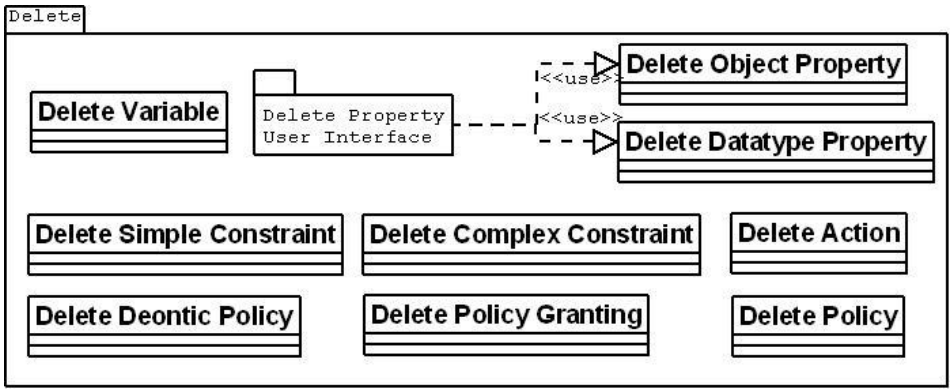
Politika *düzenleme* biriminin *yeniden adlandırma*, *düzenleme* ve *silme* birimleri de 9 sınıftan oluşmaktadır. Bu birimlerin yapısı sırası ile Şekil 7.5, 7.6 ve 7.7'de gösterilmektedir.



Şekil 7.5 Politika düzenlemesi biriminin yeniden adlandırma birimi.

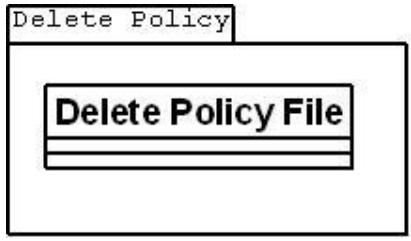


Şekil 7.6 Politika düzenlemesi biriminin düzenleme birimi.



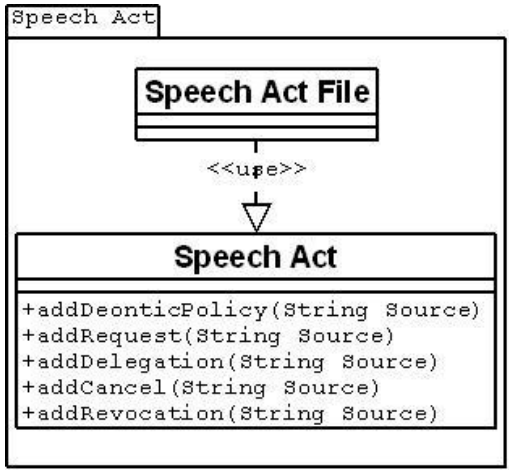
Şekil 7.7 Politika düzenlemesi biriminin silme birimi.

Şekil 7.8'de politika *silme* birimi yer almaktadır. Bu birim sadece politika dosyasının silinmesi (Delete Policy File) sınıfından oluşmaktadır.



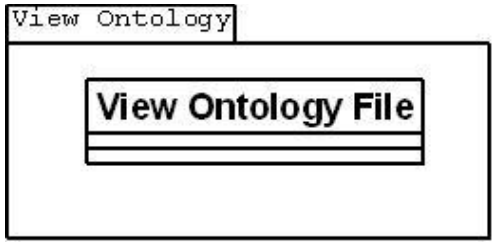
Şekil 7.8 Politika silme birimi.

Konuşma edimi biriminin yer aldığı Şekil 7.9'da konuşma edimi dosyası sınıfı konuşma edimi sınıfını kullanmaktadır. Bu sınıfta; `addDeonticPolicy`, `addRequest`, `addDelegation`, `addCancel` ve `addRevocation` işlemleri gerçekleştirilmektedir.



Şekil 7.9 Konuşma edimi birimi.

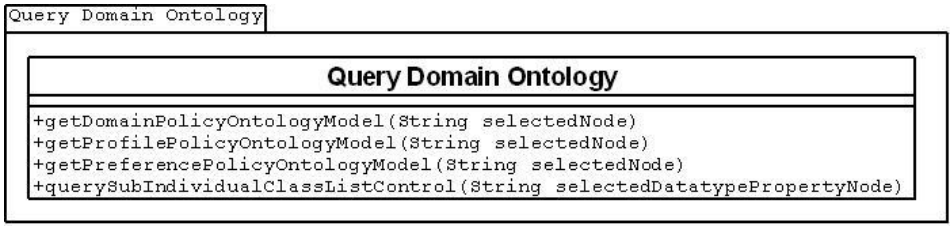
Ontoloji görüntüleme birimi Şekil 7.10'da yer almaktadır. Bu birim ontoloji dosyasının görüntülenmesi (View Ontology File) sınıfından oluşmaktadır.



Şekil 7.10 Ontoloji görüntüleme birimi.

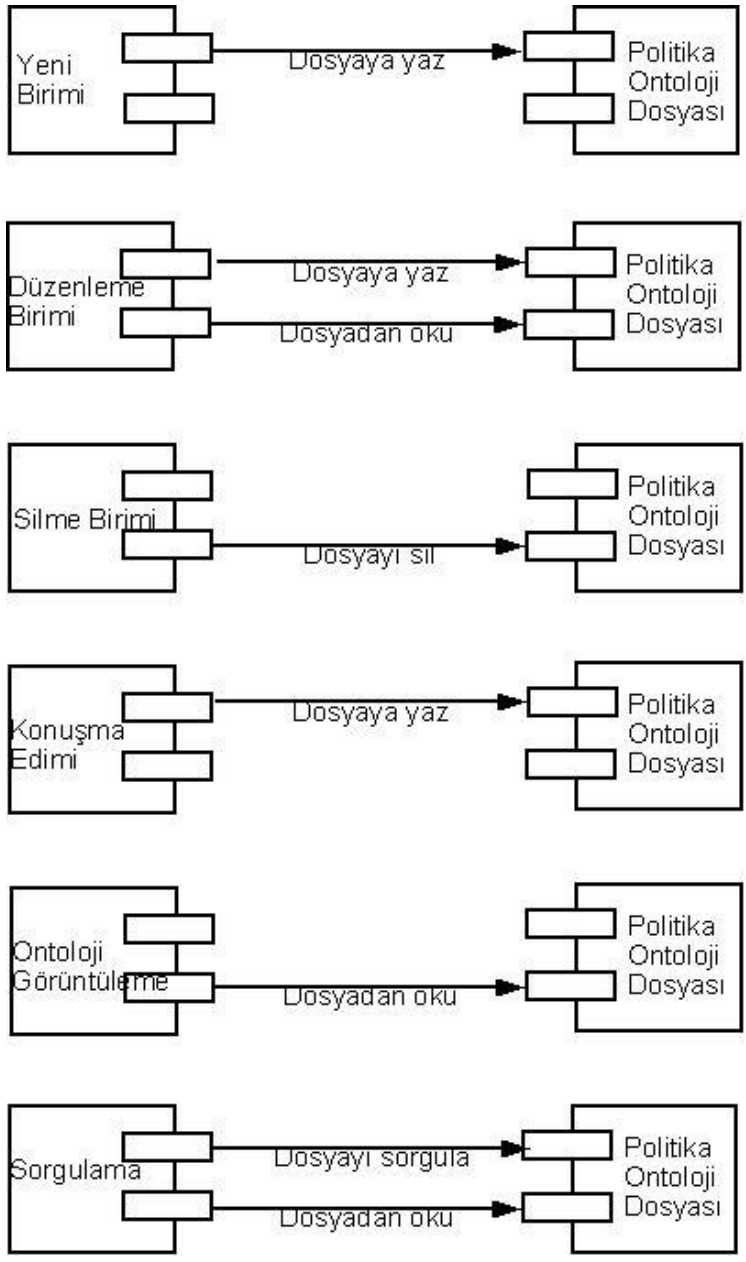
Şekil 7.11 *etki alanı ontolojisi sorgulama* birimini göstermektedir. Bu birim etki alanı ontolojisinin sorgulanması (Query Domain

Ontology) sınıfından oluşmaktadır. Bu sınıfta etki alanı, profil ve seçenek politika ontolojileri kullanılarak gerçekleştirilen sorgulamalar için gerekli olan temel işlemler; `getDomainPolicyOntologyModel`, `getProfilePolicyOntologyModel`, `getPreferencePolicy OntologyModel` ve `querySubIndividualClassListControl`'dur.



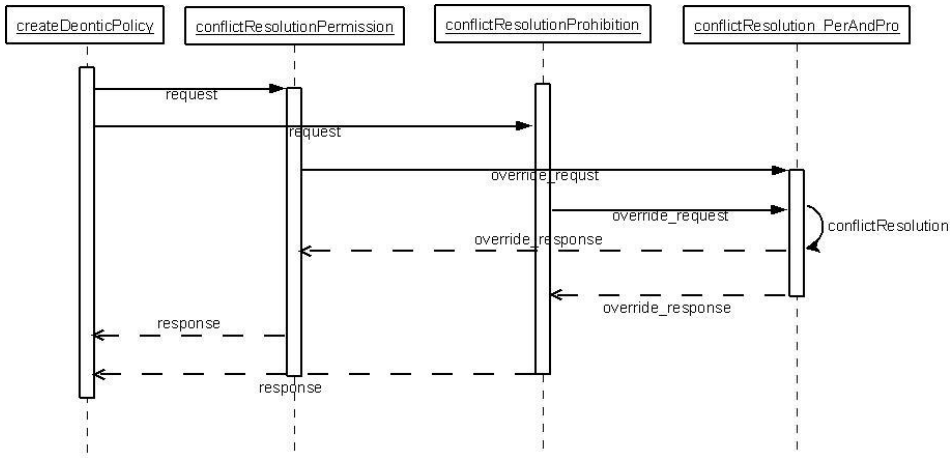
Şekil 7.11 Etki alanı ontolojisi sorgulama birimi.

Şekil 7.12'de politika motorunun yapısal paylaşılan veri görünümü (*architectural shared data style*) yer almaktadır. Burada her bir birimin politika ontoloji dosyası üzerinde gerçekleştirdiği işlemler gösterilmektedir.



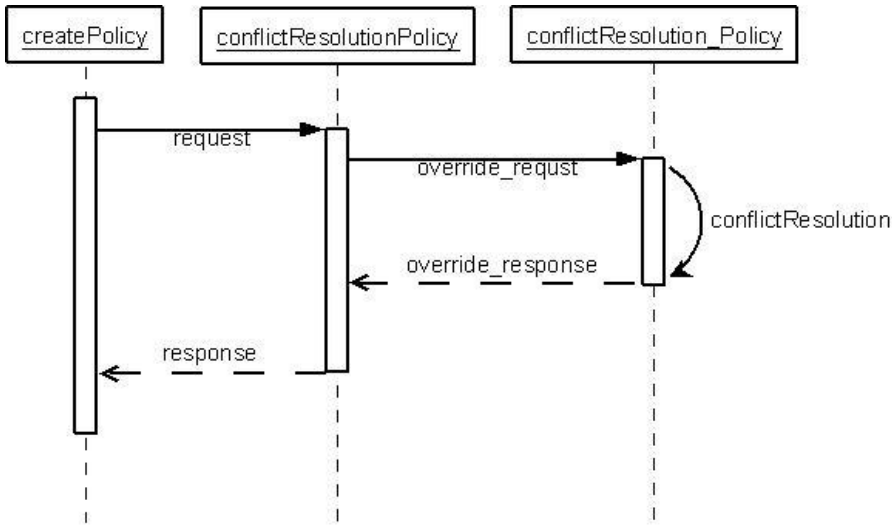
Şekil 7.12 Politika motoru yapısal paylaşılan veri görünümü.

Politika kuralının çelişmesinin çözümü için izinin yasak üzerine ya da yasağın izin üzerine önceliği kullanıcı tarafından belirtildikten sonra çelişme çözümü gerçekleştirilmektedir. Bu akış Şekil 7.13'de görülmektedir. Deontik politika oluşturulurken izin ve yasak sınıflarında kural çelişmesi olup olmadığı tespit edilmektedir. Eğer herhangi bir kural çelişmesi oluşmuş ise `conflictResolution_PerAndPro` sınıfında kullanıcıdan alınan üstünlük bilgisine göre çelişki çözülmektedir.



Şekil 7.13 Politika kuralı çelişme çözümü akış diyagramı.

Politikalar arasında oluşabilecek çelişmelerin çözümü için ilgili akış şeması Şekil 7.14'te yer almaktadır. Politika oluşturulurken, çelişme tespiti yapılıp, çelişme olması durumunda hangi politikanın önceliğe sahip olacağı kullanıcı tarafından belirtildikten sonra politika çelişmesi çözümü gerçekleştirilmektedir.



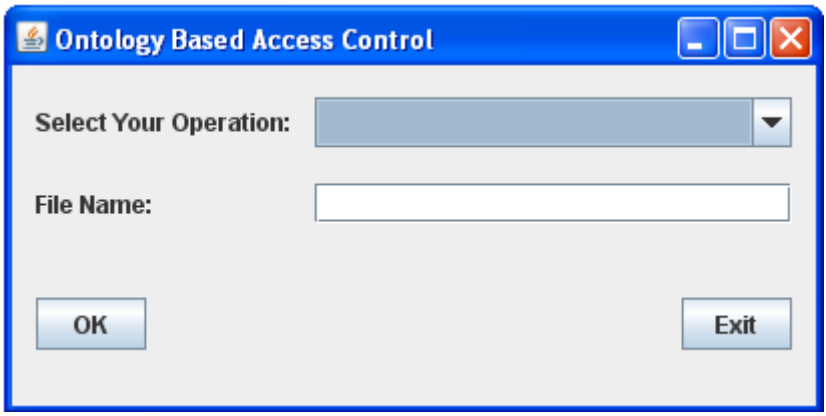
Şekil 7.14 Politika çelişmesi çözümü akış diyagramı.

7.4 Ontoloji Tabanlı Erişim Denetimi Gerçekleştirimi

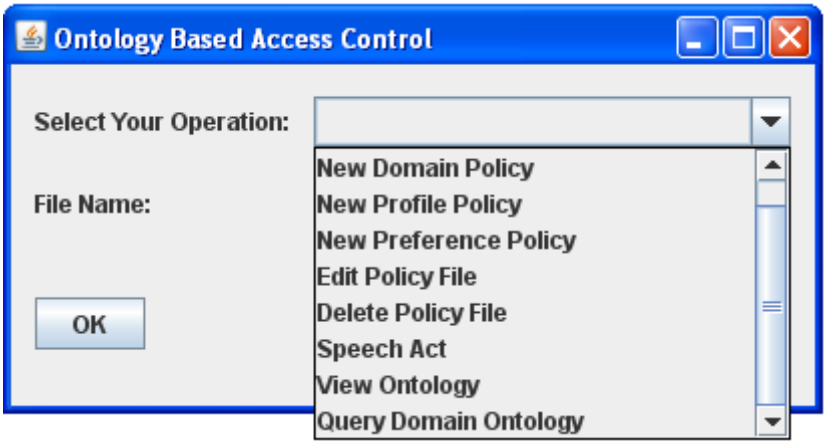
Ontoloji Tabanlı Erişim Denetimi için gerçekleştirilen uygulama çalıştırıldığında, öncelikle, kullanıcı gerçekleştirmek istediği işlemi seçmekte ve ontoloji dosyasının ismini belirtmektedir. Kullanıcının karşılaşacağı ilk ekran görüntüsü Şekil 7.15'de yer almaktadır. Şekil 7.16'da görülen kullanıcının gerçekleştirebileceği işlemler sırası ile aşağıdaki gibidir:

- Yeni Etki Alanı Politikası (New Domain Policy)
- Yeni Profil Politikası (New Profile Policy)
- Yeni Seçenek Politikası (New Preference Policy)

- Politika Dosyasının Düzenlenmesi (Edit Policy File)
- Politika Dosyasının Silinmesi (Delete Policy File)
- Konuşma Edimleri (Speech Act)
- Ontolojinin Görüntülenmesi (View Ontology)
- Etki Alanı Ontolojisinin Sorgulanması (Query Domain Ontology)



Şekil 7.15 Ontoloji tabanlı erişim denetimi arayüzü.



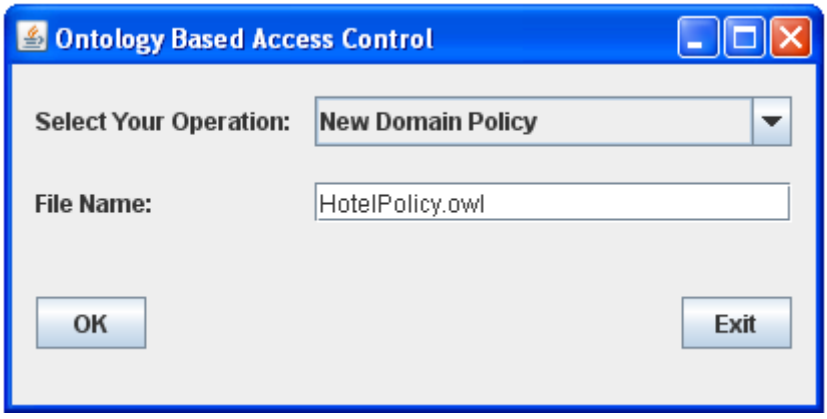
Şekil 7.16 Kullanıcının gerçekleştirebileceği işlemler.

Gerçekleştirilebilecek işlemler sırası ile izleyen bölümlerde açıklanmaktadır.

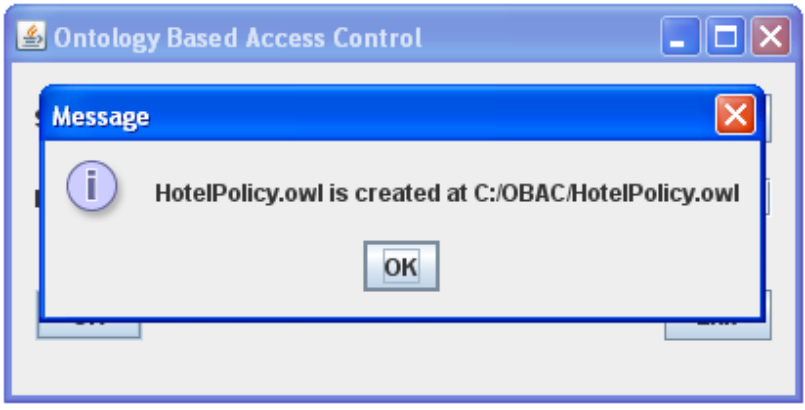
7.4.1 Yeni Etki Alanı / Profil / Seçenek politikası

Bu bölümde kullanıcının yeni etki alanı politika ontolojisi oluştururken karşılaşacağı arayüz ve işlemler anlatılmaktadır. Bu bölümde anlatılan işlemler yeni profil ve yeni seçenek politikası ontolojilerinin oluşturulmasında da aynı biçimde gerçekleşmektedir.

Yeni etki alanı politikası arayüzü aracılığı ile etki alanını temel alan yeni bir politika ontolojisi oluşturulmaktadır. Kullanıcının dosya adı kısmına girdiği yeni politika ontolojisi Şekil 7.17 ve Şekil 7.18'de görüldüğü gibi yaratılmaktadır.



Şekil 7.17 Yeni etki alanı politikasının yaratılması.

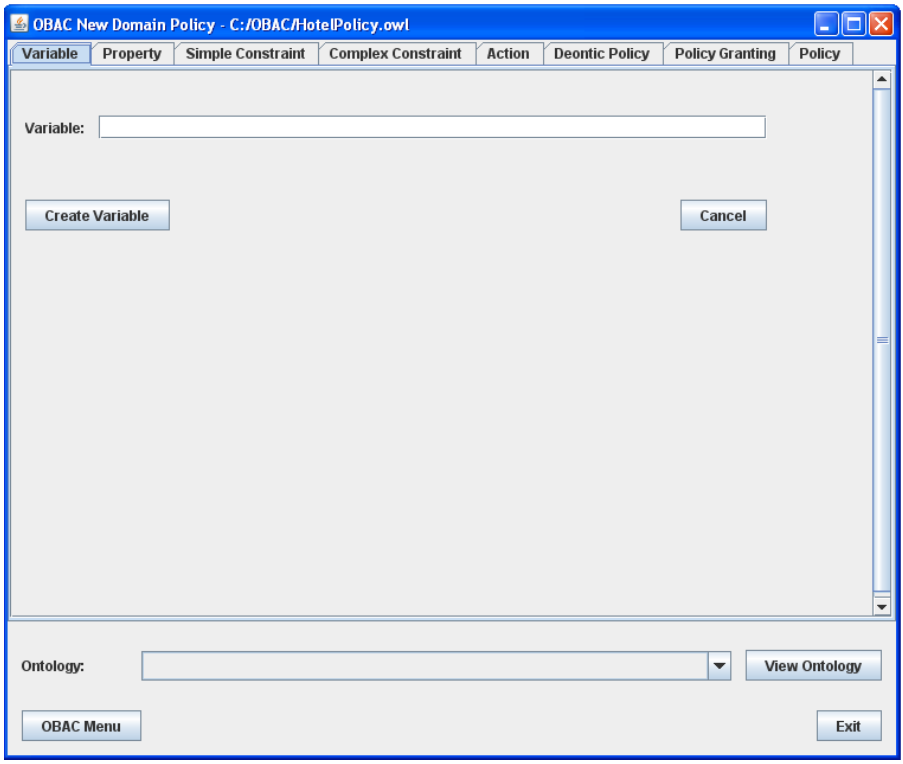


Şekil 7.18 Yeni etki alanı politikasının yaratıldığının onayı.

Yeni etki alanı politika ontoloji dosyası oluşturulduktan sonra kullanıcının karşılaşacağı ekran görüntüsü Şekil 7.19'da yer almaktadır. Bu pencerede yer alan ve politika ontolojisinin yaratılması için gerekli olan bölümler aşağıdaki gibidir:

- Değişken (Variable)
- Özellik (Property)
- Basit Kısıt (Simple Constraint)
- Karmaşık Kısıt (Complex Constraint)
- Eylem (Action)
- Deontik Politika (Deontic Policy)
- Politika Onay (Policy Granting)
- Politika (Policy)

Politika ontolojisinin oluşturulmasında kullanılan bu alanların yanı sıra, varolan tanımlanmış ontolojilerin görüntülenmesini sağlayan Ontoloji Görüntüle (View Ontology) kısmı bu pencerede yer almaktadır.



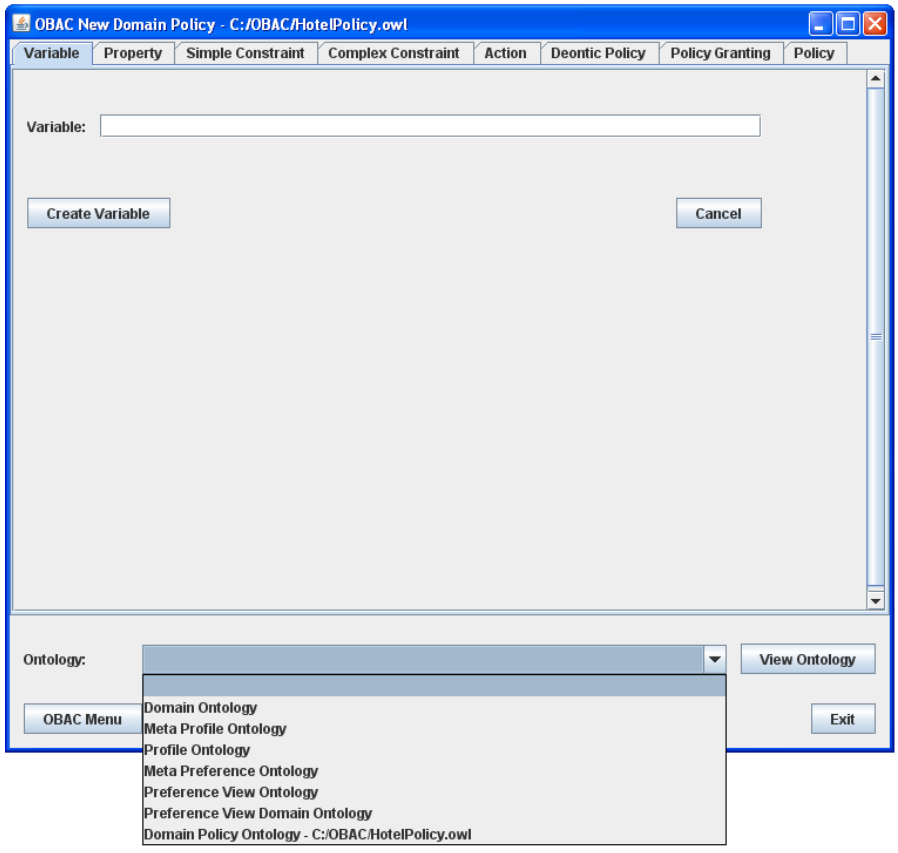
Şekil 7.19 Yeni etki alanı politikası arayüzü.

Şekil 7.20'de yer alan ontoloji görüntüleme kısmında yer alan görüntülenebilecek ontolojiler şunlardır:

- Etki Alanı Ontolojisi (Domain Ontology): Kullanılan etki alanı ontolojisini göstermektedir.
- Üst Profil Ontolojisi (Meta Profile Ontology): Profil tanımlamak için gerekli üst verilerin tanımlandığı ontolojidir.
- Profil Ontolojisi (Profile Ontology): Profil, profil adı, yaş ve meslek bilgilerinin tanımlandığı ontolojidir.

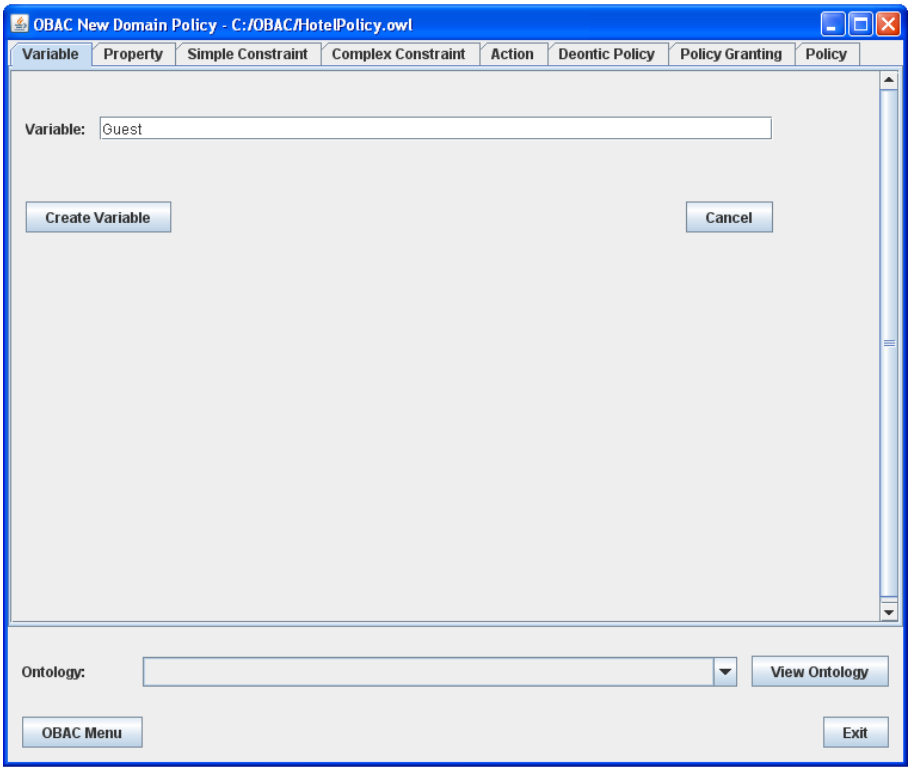
- Üst Seçenek Ontolojisi (Meta Preference Ontology): Kullanıcının kişisel seçeneklerini gösterebilmek için farklı seçenek türlerinin tanımlandığı ontolojidir.
- Seçenek Görüntüleme Ontolojisi (Preference View Domain Ontology): Seçeneklere göre ayrıştırılmış etki alanı ontolojisidir.
- Etki Alanı Seçenek Ontolojisi (PVD_Domain Ontology): PVD ile ayrıştırılan etki alanı ontolojisinin örneklerini tutan ontolojidir.
- Etki Alanı Politika Ontolojisi (Domain Policy Ontology): Üzerinde çalışılan politika ontolojisini göstermektedir.

Ontolojinin görüntülenmesi ile ilgili ayrıntılı açıklamalar 7.4.4 bölümünde yapılmaktadır.



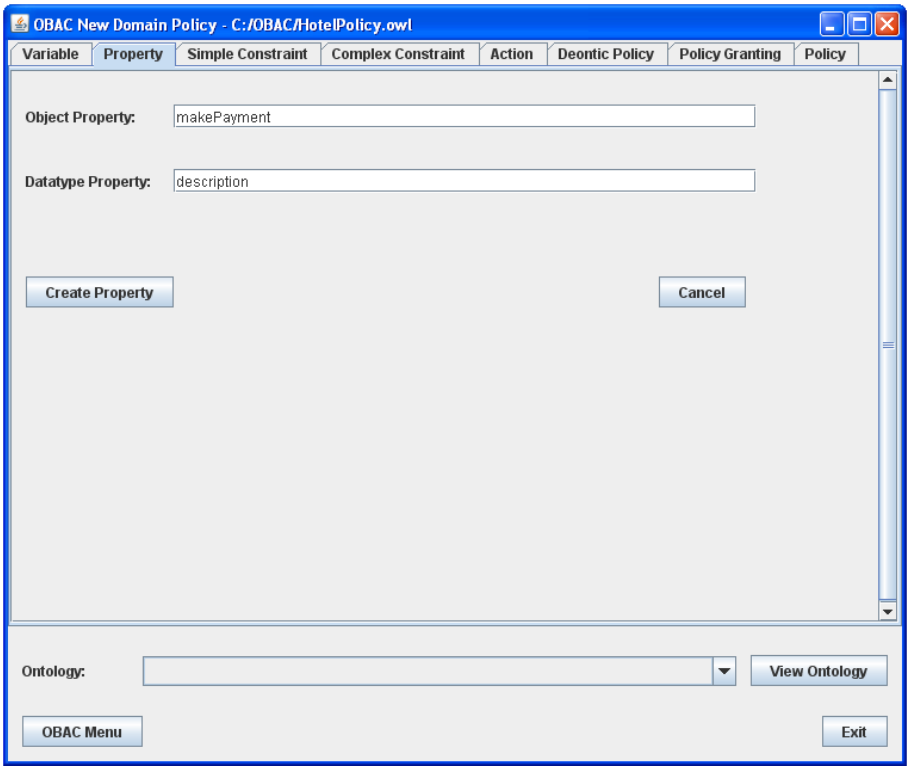
Şekil 7.20 Mevcut ontolojilerin görüntülenmesi.

Değişken tanımlamalarının yapıldığı Şekil 7.21'de yer alan *Variable* sekmesinde yaratılacak değişkenin isminin yazıldığı metin alanı yer almaktadır.



Şekil 7.21 Değişken tanımlamasının gerçekleştirimi.

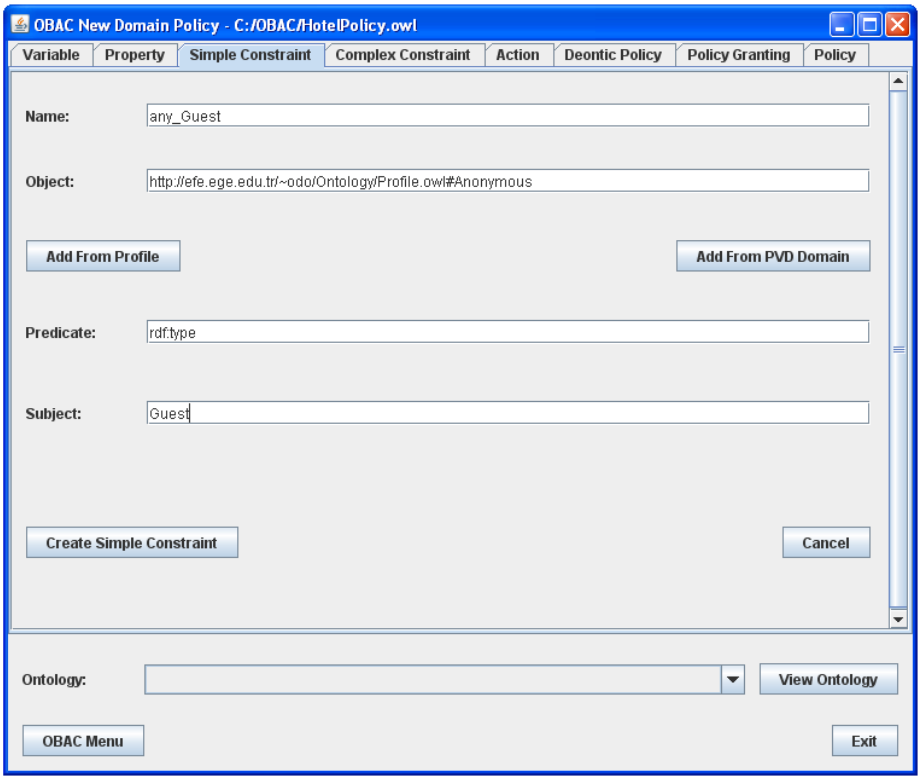
Özellik tanımlarının yapıldığı *Property* sekmesinde ise Nesne ve Veri türü olmak üzere iki özellik tanımının yapıldığı metin alanları Şekil 7.22'de gösterilmiştir.



Şekil 7.22 Özellik tanımlamalarının gerçekleştirimi.

Basit Kısıt tanımlarının yapıldığı Şekil 7.23'de yer alan *Simple Constraint* sekmesinde 4 metin alanı yer almaktadır:

- Yaratılacak basit kısıtın adı
- Nesne
    - Nesne, Profil ya da Etki Alanı Seçenek ontolojilerinden eklenebilir.
- Yüklem
- Özne

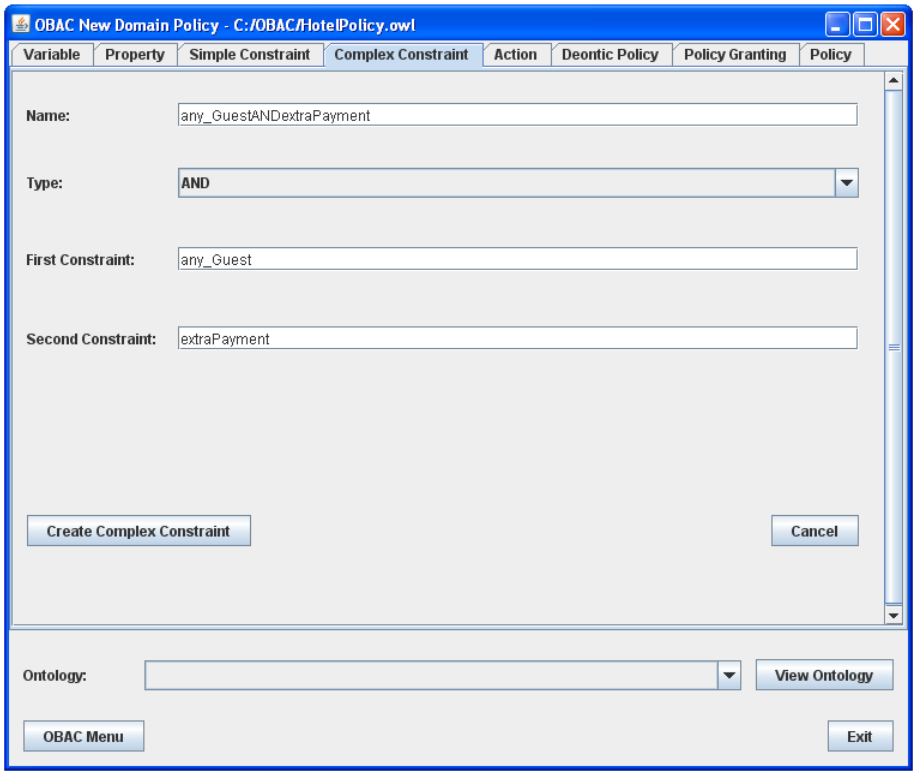


Şekil 7.23 Basit kısıt tanımının gerçekleştirimi.

Şekil 7.24'de yer alan Karmaşık Kısıt tanımlarının yapıldığı *Complex Constraint* sekmesinde de 4 metin alanı yer almaktadır:

- Yaratılacak olan karmaşık kısıdın adı
- Türü
  - AND
  - OR
  - NOT

- İlk kısıt
- İkinci kısıt



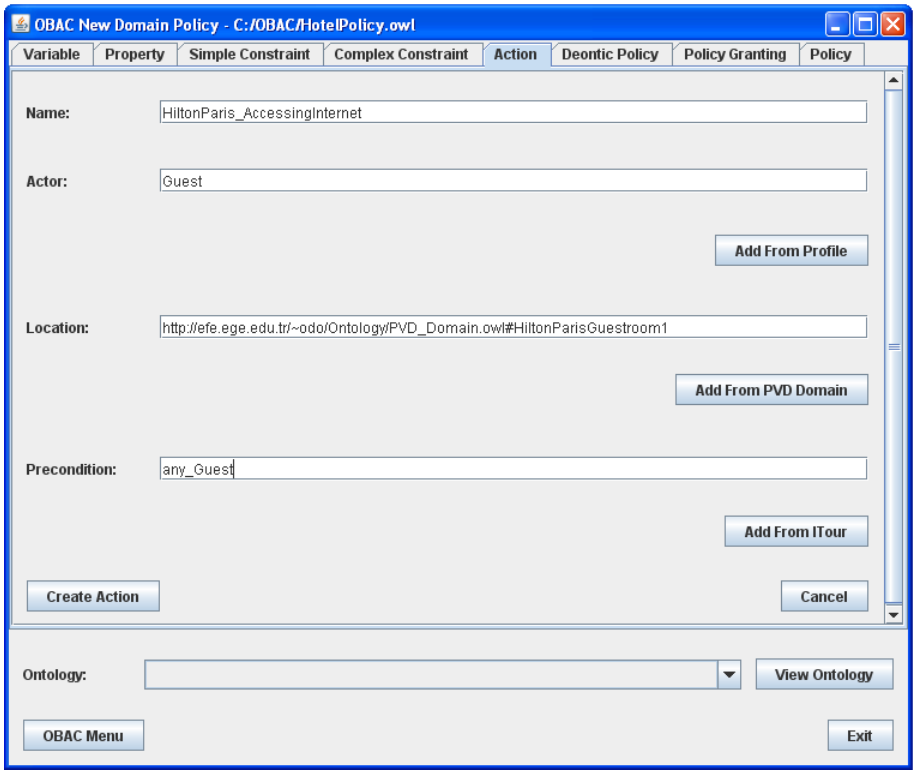
Şekil 7.24 Karmaşık kısıt tanımının gerçekleştirimi.

Action sekmesinde yapılan Eylem tanımlaması Şekil 7.25'de görülmektedir. Eylem tanımlamasında 4 metin alanı yer almaktadır:

- Eylemin adı
- Aktör

- Aktör, üzerinde çalışılan ontolojiden eklenebildiği gibi profil ontolojisinden de eklenebilmektedir.

- Konum
  - Konum, etki alanı örnekleri Etki Alanı Seçenek ontolojisinden tutulduğundan bu ontolojiden eklenebilmektedir.
- Önkoşul
  - Önkoşul e-turizm ontolojisinden eklenebilmektedir.



Şekil 7.25 Eylem tanımının gerçekleştirimi.

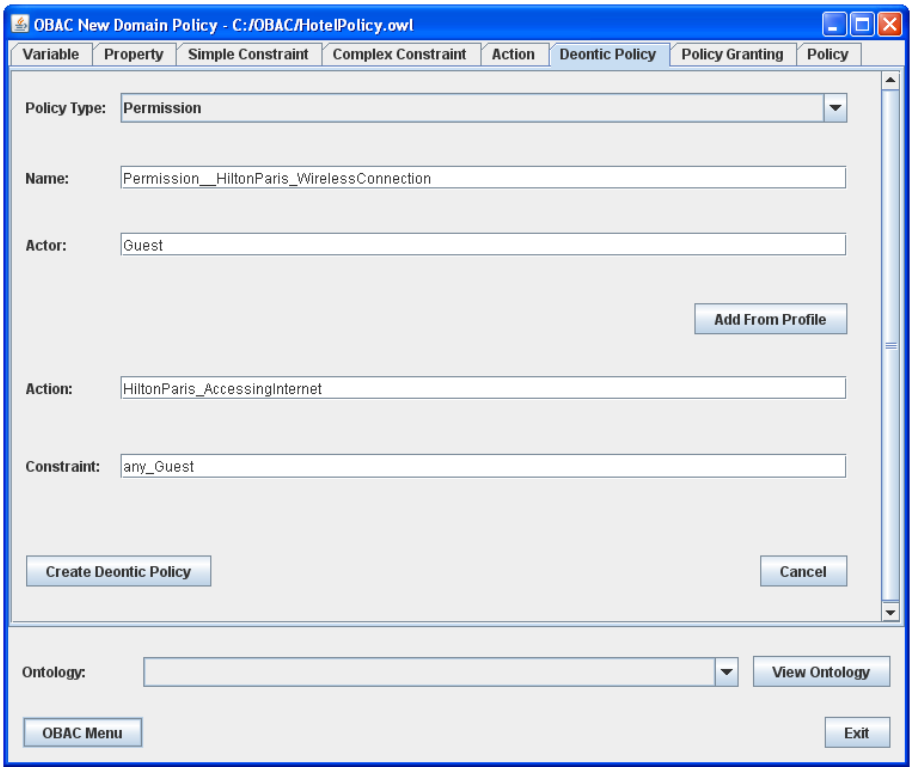
Şekil 7.26'da görülen Deontik Politika tanımlamasının yapıldığı *Deontic Policy* sekmesinde bir birleşik giriş kutusu ve 4 metin alanı yer almaktadır. Bunlar:

- Politika türü
  - İzin
  - Yasak
  - Zorunluluk
  - Özel izin

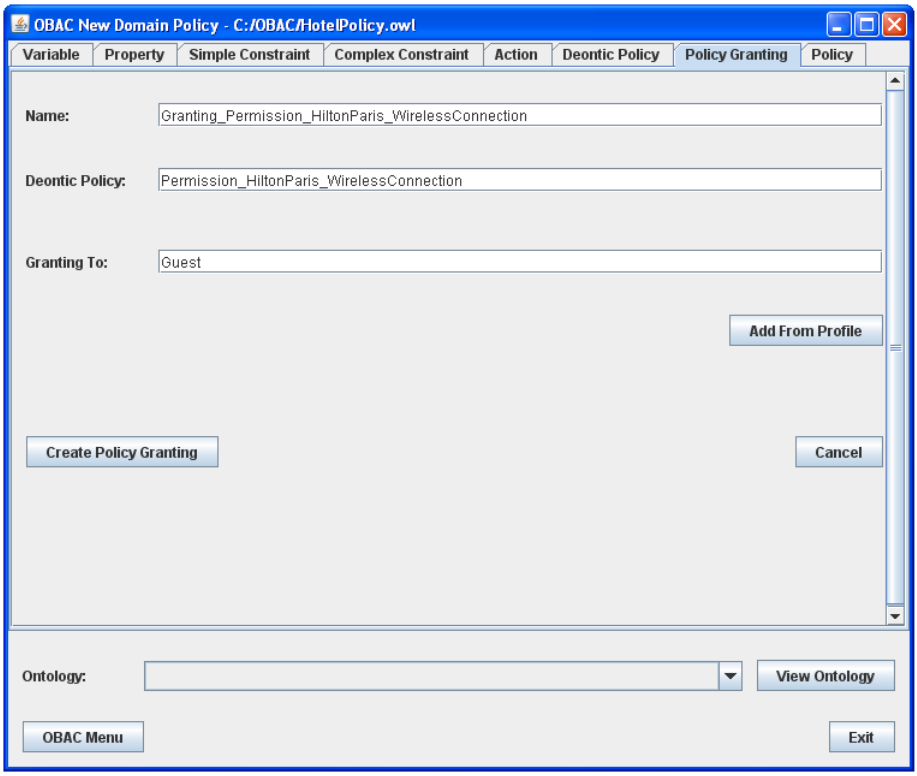
- Politikanın adı
- Aktör
    - Aktör, profil ontolojisinden eklenebilmektedir.
- Eylem
- Koşul

Politika Onay tanımının yapıldığı *Policy Granting* sekmesi Şekil 7.27'de yer almaktadır. Burada 3 metin alanı bulunmaktadır:

- Politika onayının adı
- Onaylanacak deontik politikanın adı
- Onaylamanın kime yapılacağı
    - Onaylamanın kime yapılacağı üzerinde çalışılan ontolojiden eklenebildiği gibi profil ontolojisindende eklenebilmektedir.



Şekil 7.26 Deontik politika tanımının gerçekleştirimi.

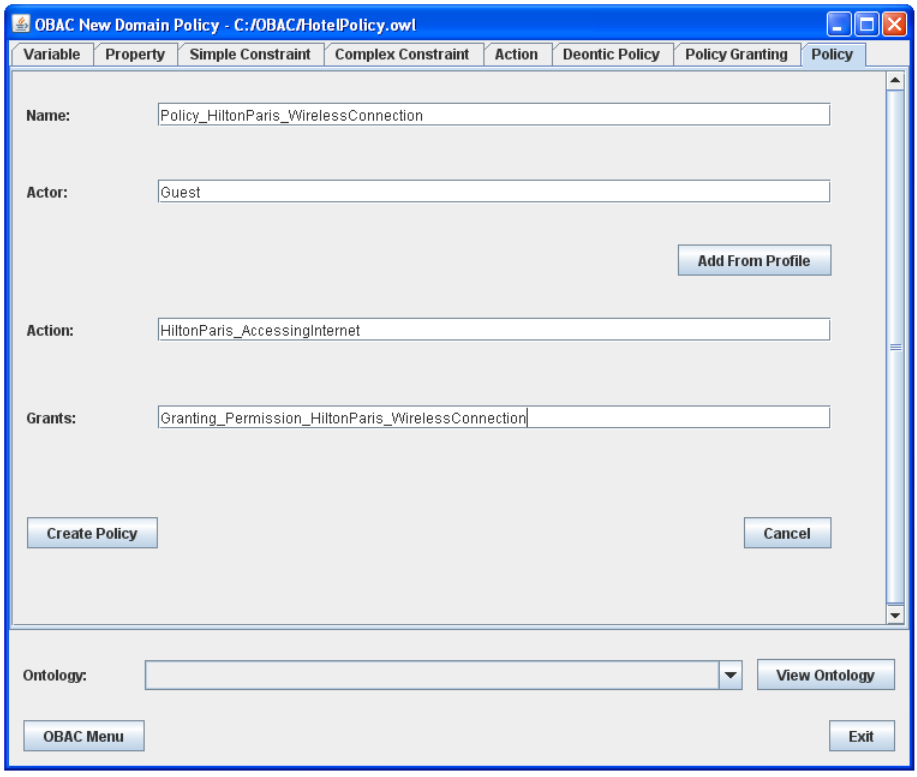


Şekil 7.27 Politika onay tanımının gerçekleştirimi.

Politika tanımlamasının son işlemi olan Politika tanımı *Policy* sekmesinde gerçekleştirilmektedir. Şekil 7.28'de yer alan bu tanımlamada 4 metin alanı bulunmaktadır:

- Politikanın adı
- Aktör
    - Politika aktörü profil ontolojisinden eklenebilmektedir.
- Eylem

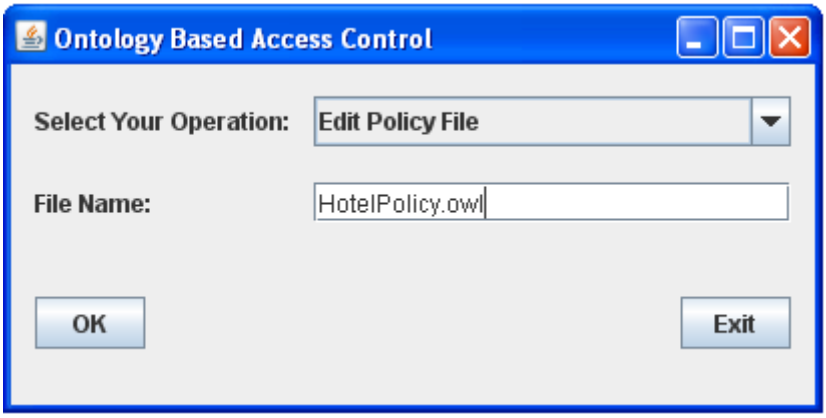
- Onaylanan politikanın adı



Şekil 7.28 Politika tanımının gerçekleştirimi.

7.4.2 Politika dosyasının düzenlenmesi

Sistemde yer alan politika ontolojilerinin üzerinde değişiklik yapılabilmesini sağlamaktadır. Düzenleme işlemi için kullanıcı öncelikle üzerinde çalışacağı politika ontoloji dosyasının adını Şekil 7.29'da görüldüğü gibi belirtmektedir.



Şekil 7.29 Politika dosyasının düzenlenmesi.

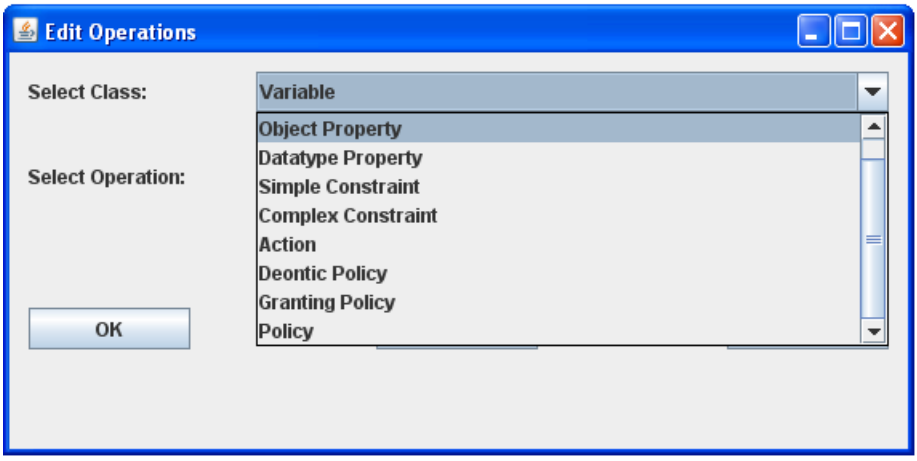
Şekil 7.30'da görülen politika ontolojisinde yer alan ve üzerinde değişiklik yapılabilecek olan ontoloji sınıfları şunlardır:

- Değişken (Variable)
- Nesne Özelliği (Object Property)
- Veri türü Özelliği (Datatype Property)
- Basit Kısıt (Simple Constraint)

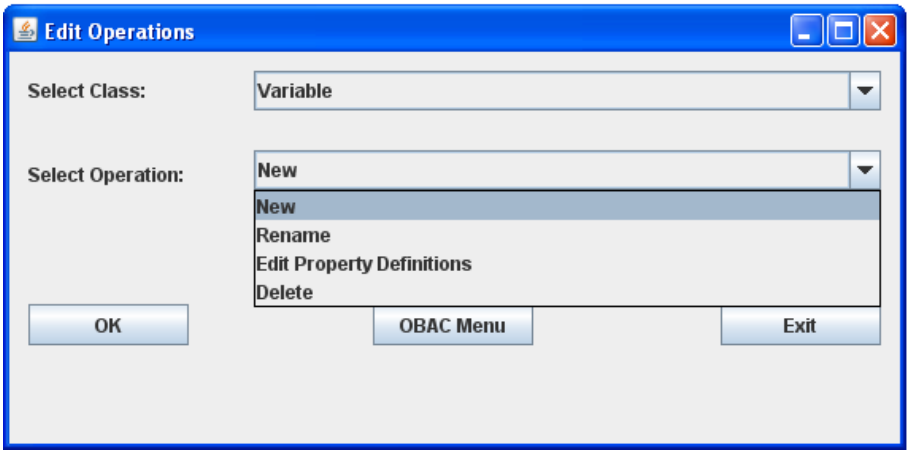
- Karmaşık Kısıt (Complex Constraint)
- Eylem (Action)
- Deontik Politika (Deontic Policy)
- Politikanın Onaylanması (Granting Policy)
- Politika (Policy)

Bu sınıfların her biri için gerçekleştirilebilecek işlemler ise Şekil 7.31'de de gösterilmiş olan aşağıdaki işlemlerdir:

- Yeni (New)
- Yeniden adlandırma (Rename)
- Özellik Tanımlarının Düzenlenmesi (Edit Property Definitions)
- Sil (Delete)

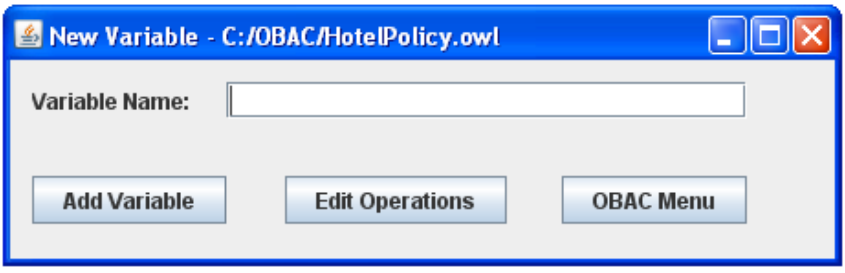


Şekil 7.30 Düzenleme işleminin yapılabileceği politika ontolojisi sınıfları.

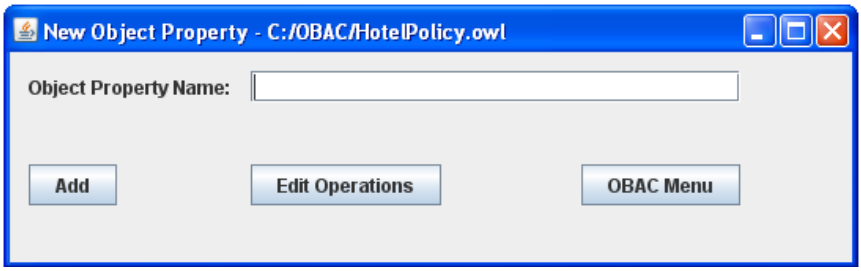


Şekil 7.31 Sınıfların düzenlenmesinde yapılabilecek işlemler.

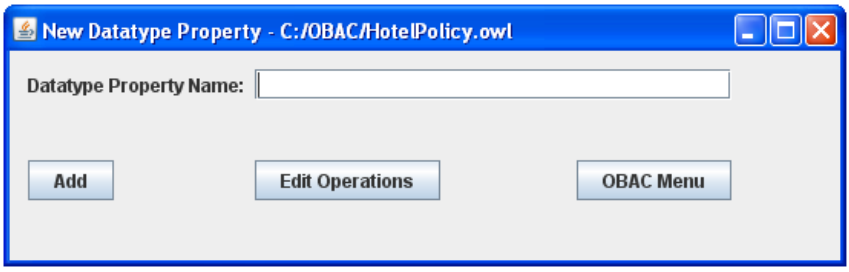
Yeni işlemi, ilgili sınıf için yeni bir tanımlama yapılmasını göstermektedir. Şekil 7.32'de de yeni bir değişken tanımı gösterilmiştir. Şekil 7.33 ve Şekil 7.34 sırası ile yeni bir nesne ve veri türü özelliği tanımlamalarını göstermektedir.



Şekil 7.32 Yeni bir değişken tanımı.



Şekil 7.33 Yeni bir nesne özelliği tanımı.

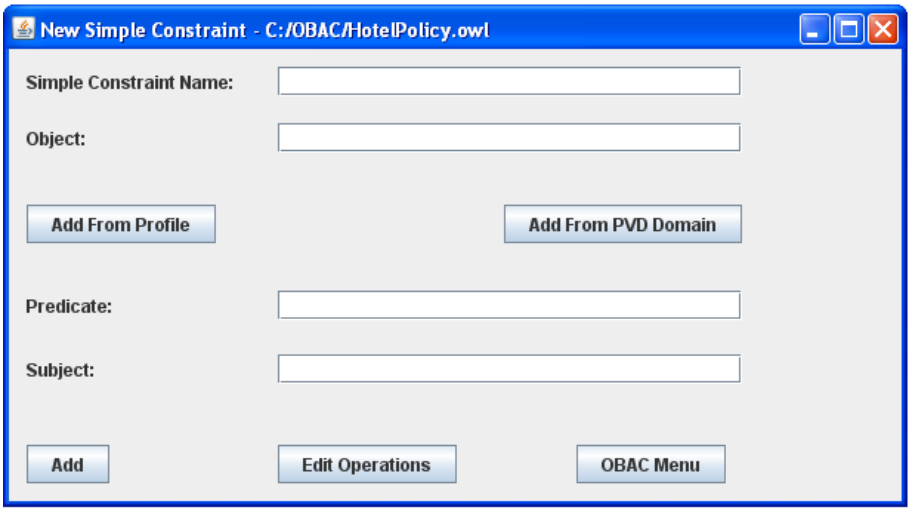


Şekil 7.34 Yeni bir veri türü özelliği tanımı.

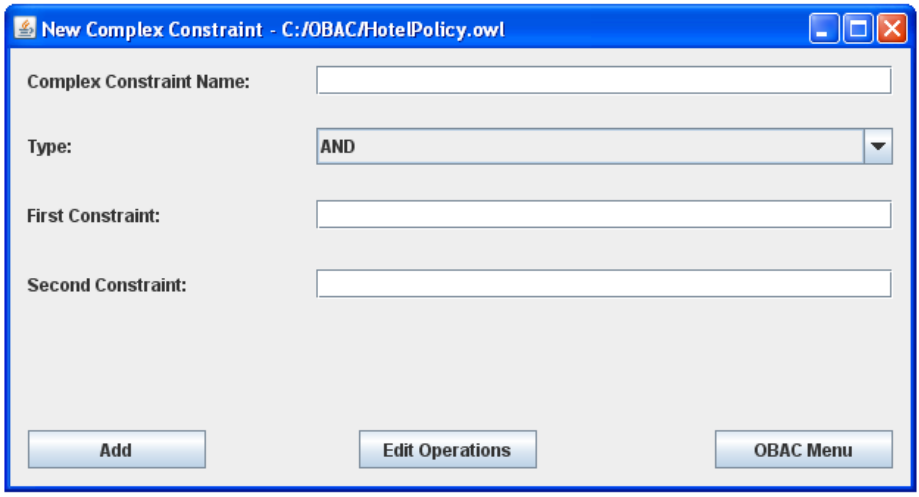
Düzenleme işlemleri pencerelerinde 3 düğme bulunmaktadır:

- Ekleme – Add
  - Yapılan tanımlama ontoloji dosyasına eklenmektedir.
- Düzenleme İşlemleri – Edit Operations
  - Şekil 7.30 ve Şekil 7.31'de yer alan Düzenleme İşlemleri penceresine dönülmektedir.
- OBAC Menü – OBAC Menu
  - Şekil 7.29'da yer alan OBAC Menüsüne dönülmektedir.

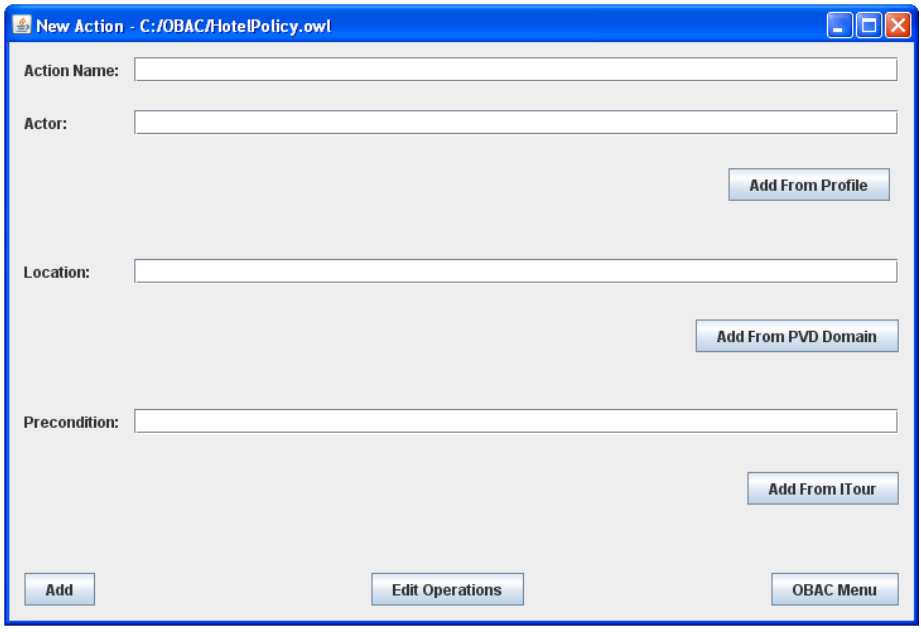
Yeni bir basit kısıt, karmaşık kısıt, eylem, deontik politika, politikanın onayı ve politika tanımları sırası ile Şekil 7.35, Şekil 7.36, Şekil 7.37, Şekil 7.38, Şekil 7.39 ve Şekil 7.40'da görülmektedir.



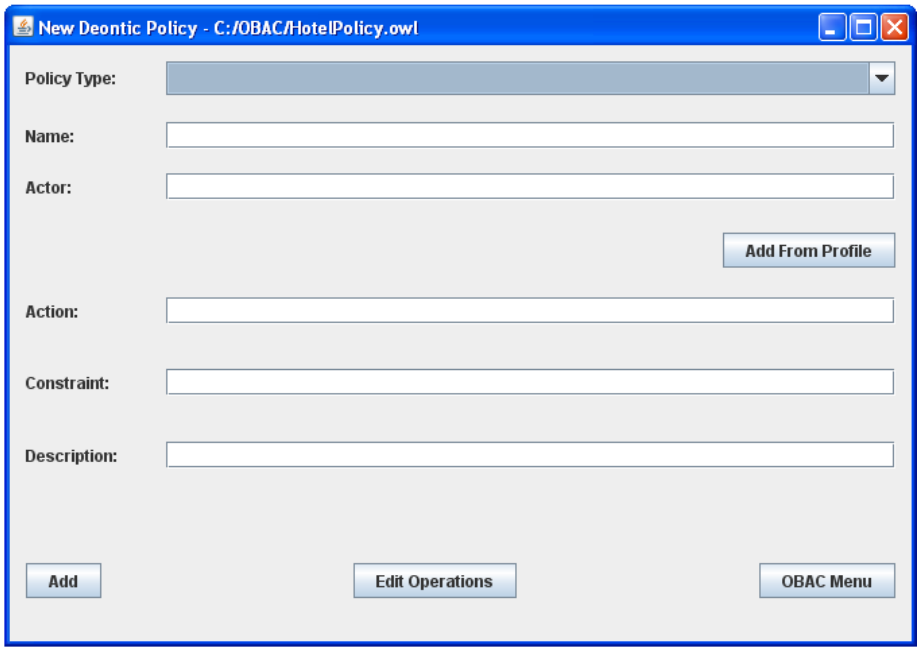
Şekil 7.35 Yeni bir basit kısıt tanımı.



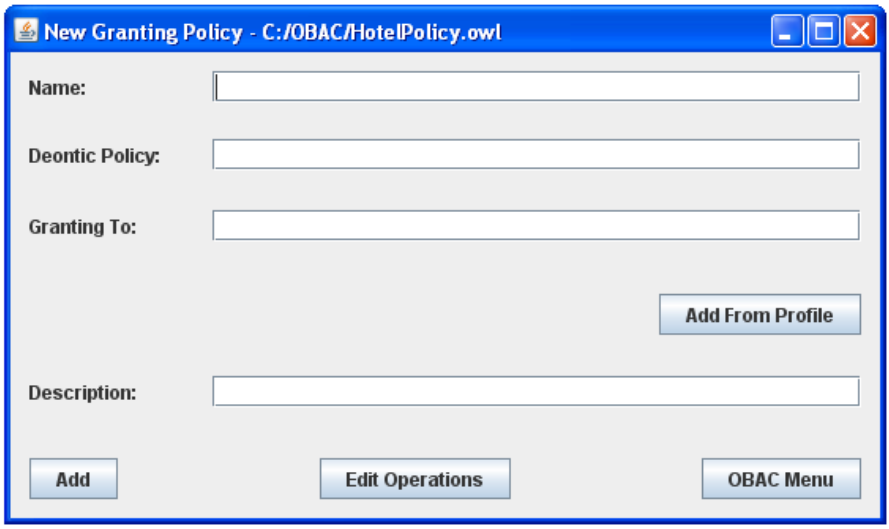
Şekil 7.36 Yeni bir karmaşık kısıt tanımı.



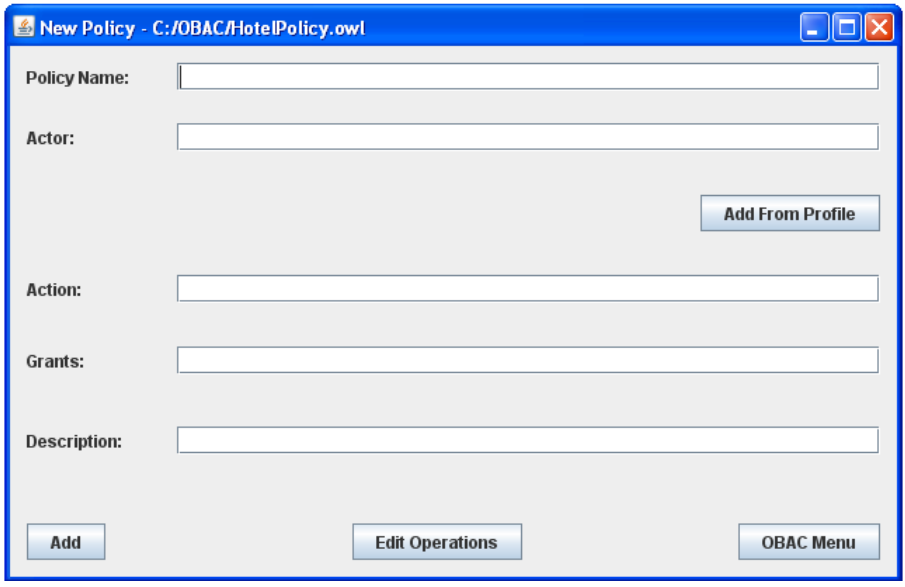
Şekil 7.37 Yeni bir eylem tanımı.



Şekil 7.38 Yeni bir deontik politika tanımı.



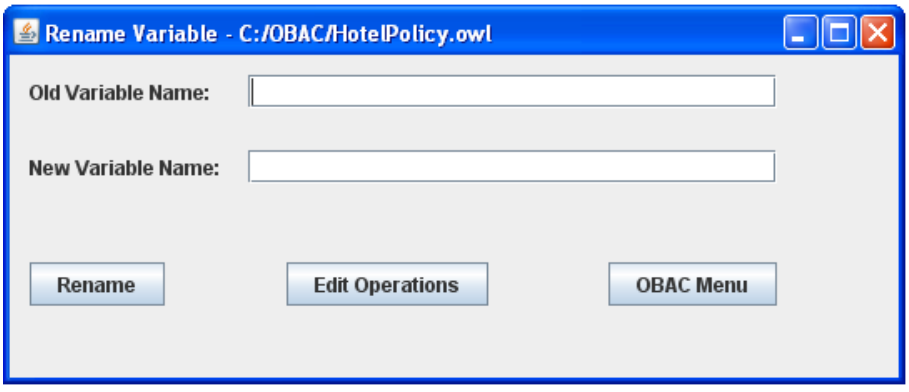
Şekil 7.39 Yeni bir politika onay tanımı.



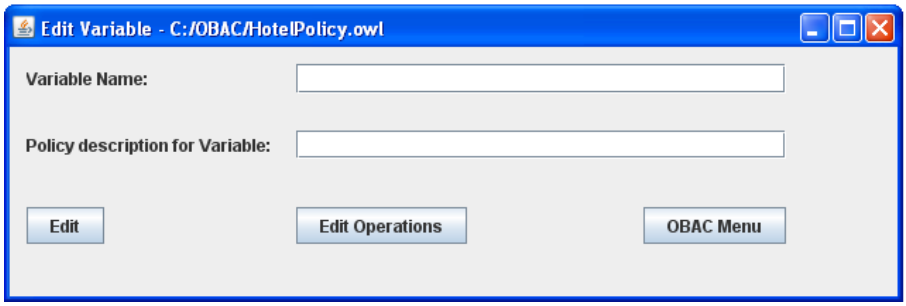
Şekil 7.40 Yeni bir politika tanımı.

Yeniden adlandırma işleminde önce ilgili sınıf örneğinin değiştirilmek istenen varolan adı daha sonra yeni adı girilmektedir. Bütün sınıf örnekleri için yeniden adlandırma işlemi penceresi aynı olduğundan bu tezde sadece Değişken için yeniden adlandırma ekran görüntüsü Şekil 7.41'de verilmiştir.

Her bir sınıfın örneği ile ilgili düzenlemelerin yapılması Özellik Tanımlarının Düzenlenmesi adı altında gerçekleştirilmektedir. Şekil 7.42'de değişken sınıfındaki bir örnek ile ilgili yapılabilecek özellik tanımlaması görülmektedir. Burada, değişken sınıfındaki özelliklere politika içindeki tanımlarını belirten bir tanımlama özelliği eklenmekte ya da varolan tanımlamaları güncellenmektedir. Nesne ve veri türü özelliklerinde de özellik tanımlaması düzenlemesi değişken sınıfındaki gibi gerçekleştirilmektedir.

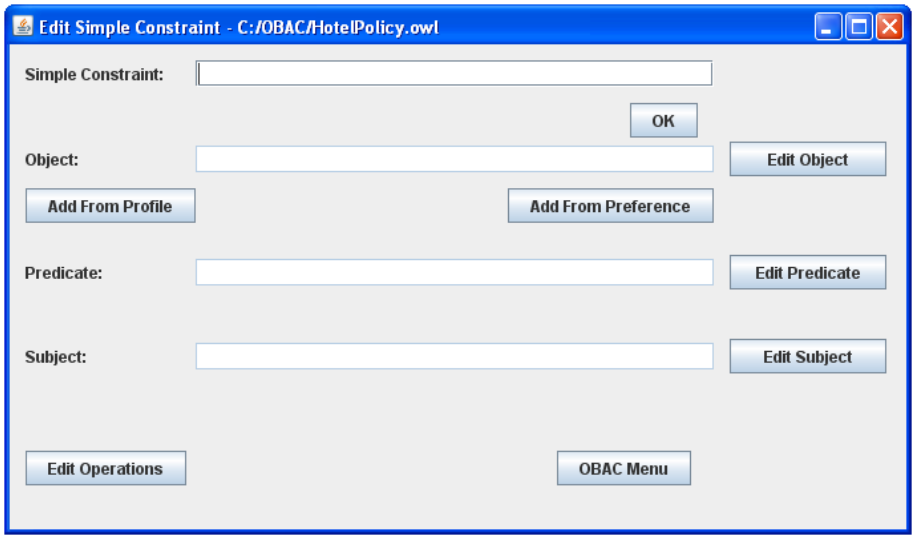


Şekil 7.41 Değişken için yeniden adlandırma.

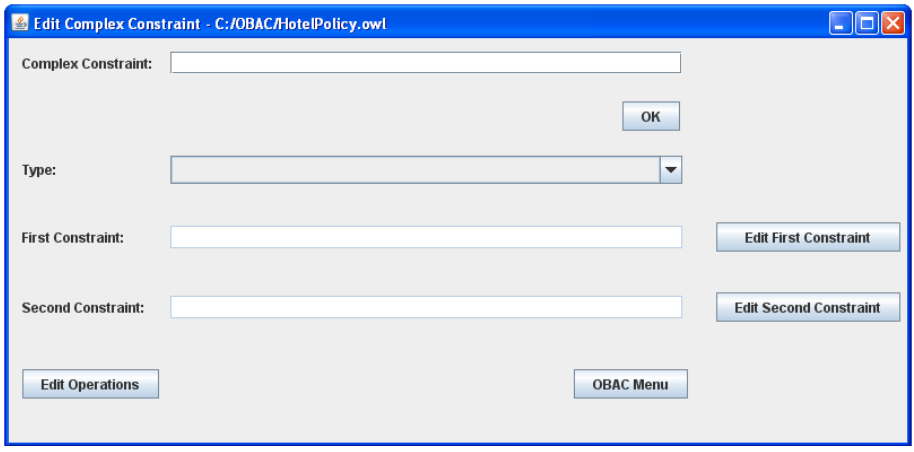


Şekil 7.42 Değişken için özellik tanımlamasının düzenlenmesi.

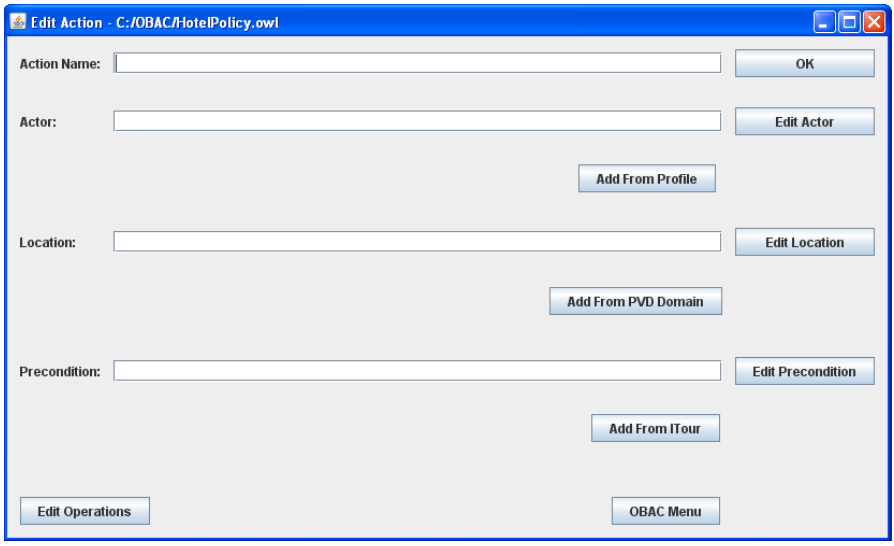
Şekil 7.43, Şekil 7.44, Şekil 7.45, Şekil 7.46, Şekil 7.47 ve Şekil 7.48 sırası ile basit kısıt, karmaşık kısıt, eylem, deontik politika, politika onayı ve politika örnekleri için özellik tanımlaması düzenlenmesini göstermektedir. Bu şekillerdeki ortak nokta; öncelikle örnek adı girilmesi, OK düğmesine basıldıktan sonra üzerinde değişiklik yapılabilecek olan özellikler ile ilgili metin alanlarının seçilemez durumdan seçilebilir duruma geçmesidir. OK düğmesine basıldığında ilgili örneğin ontoloji dosyasındaki özellik değerleri ilgili metin alanlarında görüntülenmektedir. Kullanıcı değişiklik yapmak istediği alanları düzenleyip her bir alanın karşısında yer alan Düzenleme – Edit düğmesine basmalıdır.



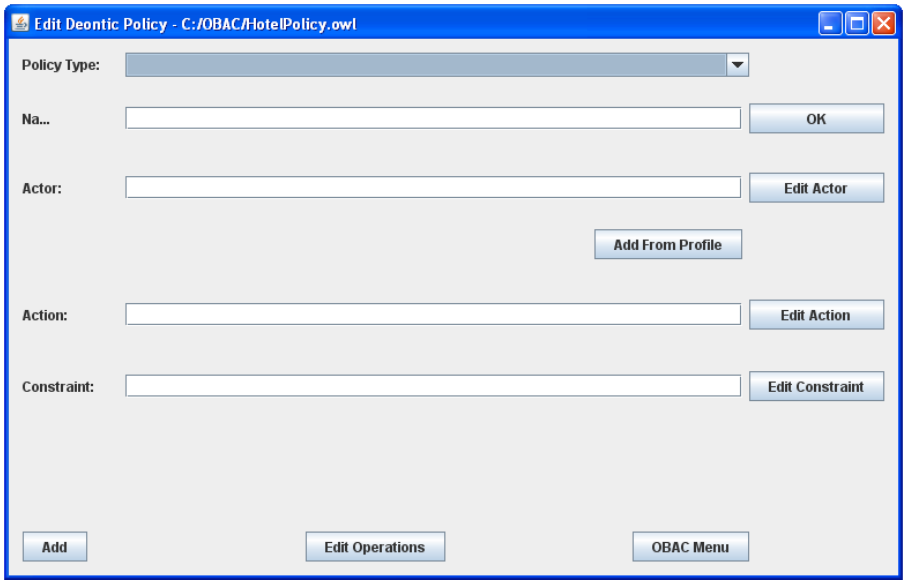
Şekil 7.43 Basit kısıt için özellik tanımlamasının düzenlenmesi.



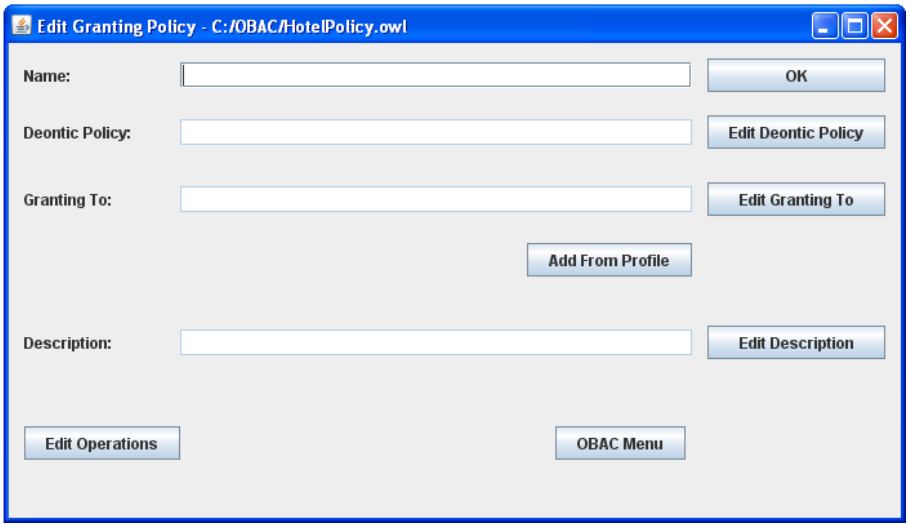
Şekil 7.44 Karmaşık kısıt için özellik tanımlamasının düzenlenmesi.



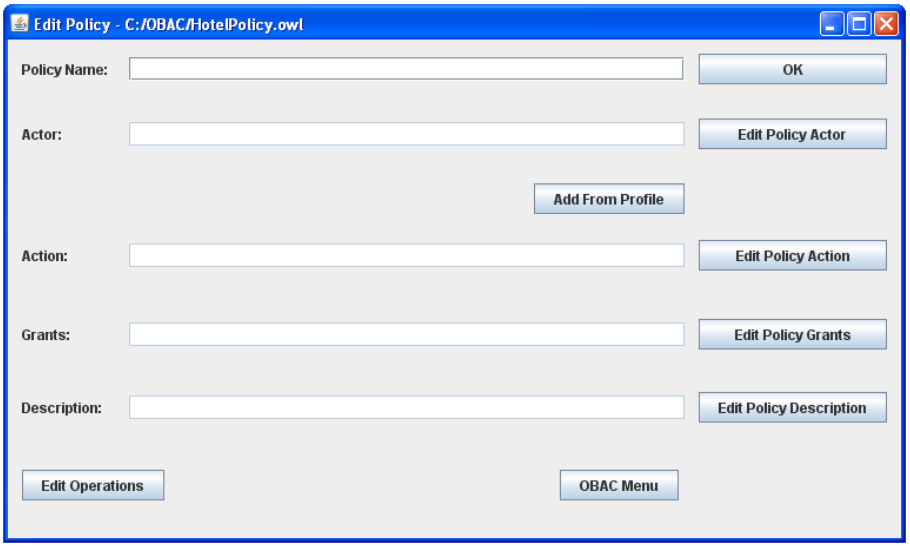
Şekil 7.45 Eylem için özellik tanımlamasının düzenlenmesi.



Şekil 7.46 Deontik politika için özellik tanımlamasının düzenlenmesi.

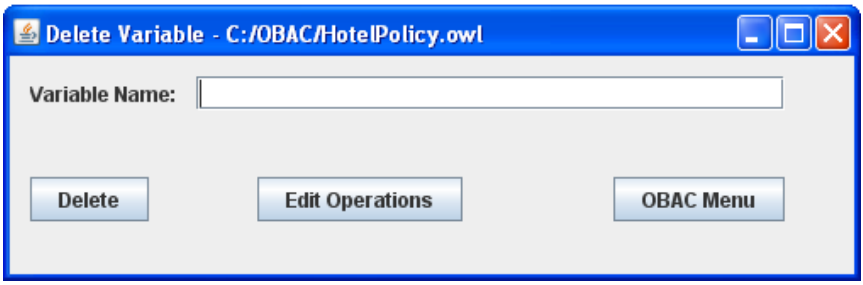


Şekil 7.47 Politika onayı için özellik tanımlamasının düzenlenmesi.



Şekil 7.48 Politika için özellik tanımlamasının düzenlenmesi.

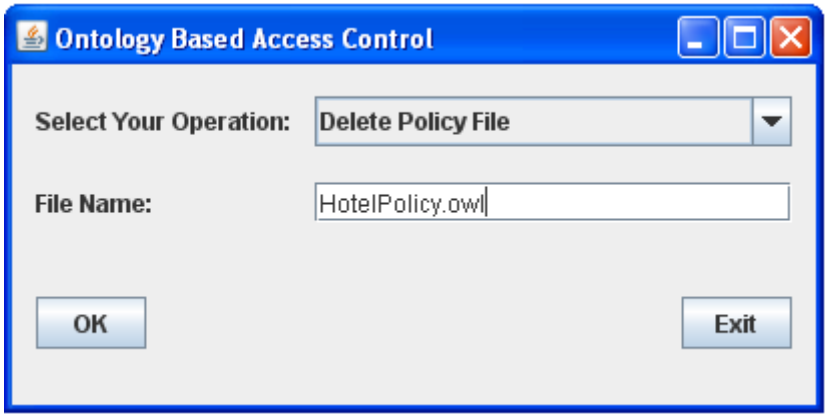
Ontoloji sınıf örneğinin silinmesi işlemi bütün sınıflar için aynı şekilde gerçekleşmektedir. Şekil 7.49, değişken sınıfında yer alan bir örneğin silinmesi işlemi sırasında kullanıcının karşılaşacağı ekran görüntüsünü göstermektedir.



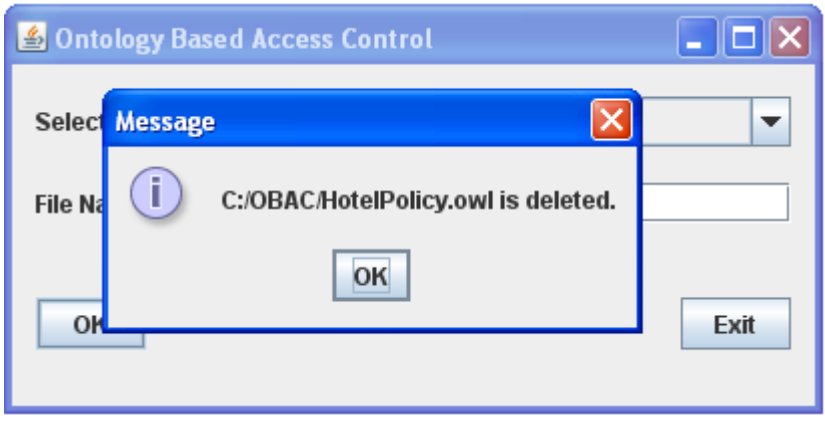
Şekil 7.49 Değişken sınıfındaki bir örneğin silinmesi.

7.4.3 Politika dosyasının silinmesi

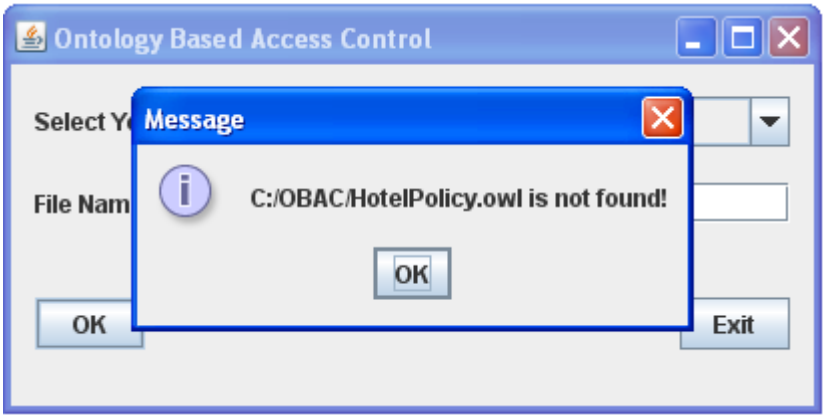
Dizgede yer alan politika ontolojisinin silinmesi için kullanıcı Şekil 7.50'de görüldüğü gibi dosya adını belirtmektedir. Belirtilen politika ontolojisi sistemde yer alıyor ve başarılı bir şekilde silme işlemi gerçekleştiyse Şekil 7.51'de belirtilen dosya silindi iletisi, belirtilen dosya sistemde bulunamadı ve silme işlemi gerçekleştirilemediği durumda ise Şekil 7.52'de belirtilen dosya bulunamadı iletisi görüntülenecektir.



Şekil 7.50 Politika ontolojisinin silinmesi.



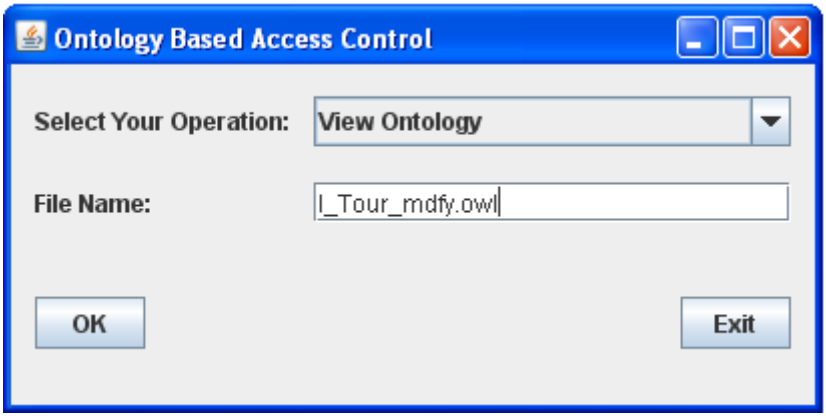
Şekil 7.51 Politika ontolojisinin başarılı bir şekilde silinmesi.



Şekil 7.52 Silinecek bir politika ontolojisinin bulunamadığının belirtilmesi.

7.4.4 Ontolojilerin görüntülenmesi

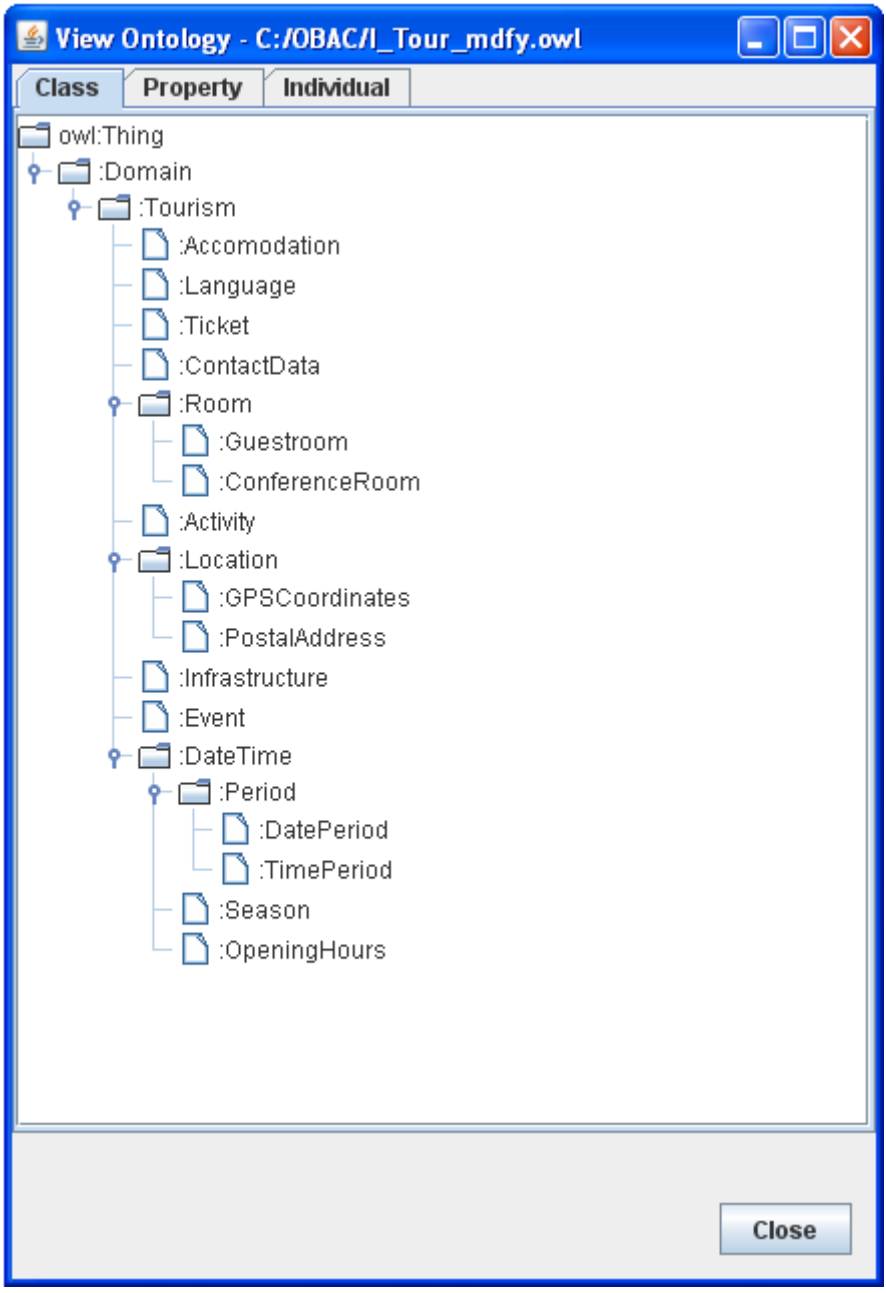
Sistemde yer alan bir ontolojinin sınıf, özellik ve örnek sıradüzensellerinin görüntülenmesi bu bölümde gerçekleşmektedir. Şekil 7.53'de görüldüğü gibi kullanıcı Ontoloji Görüntüle – View Ontology işlemini seçtikten sonra görüntülemek istediği ontoloji dosyasının adını belirtmektedir.



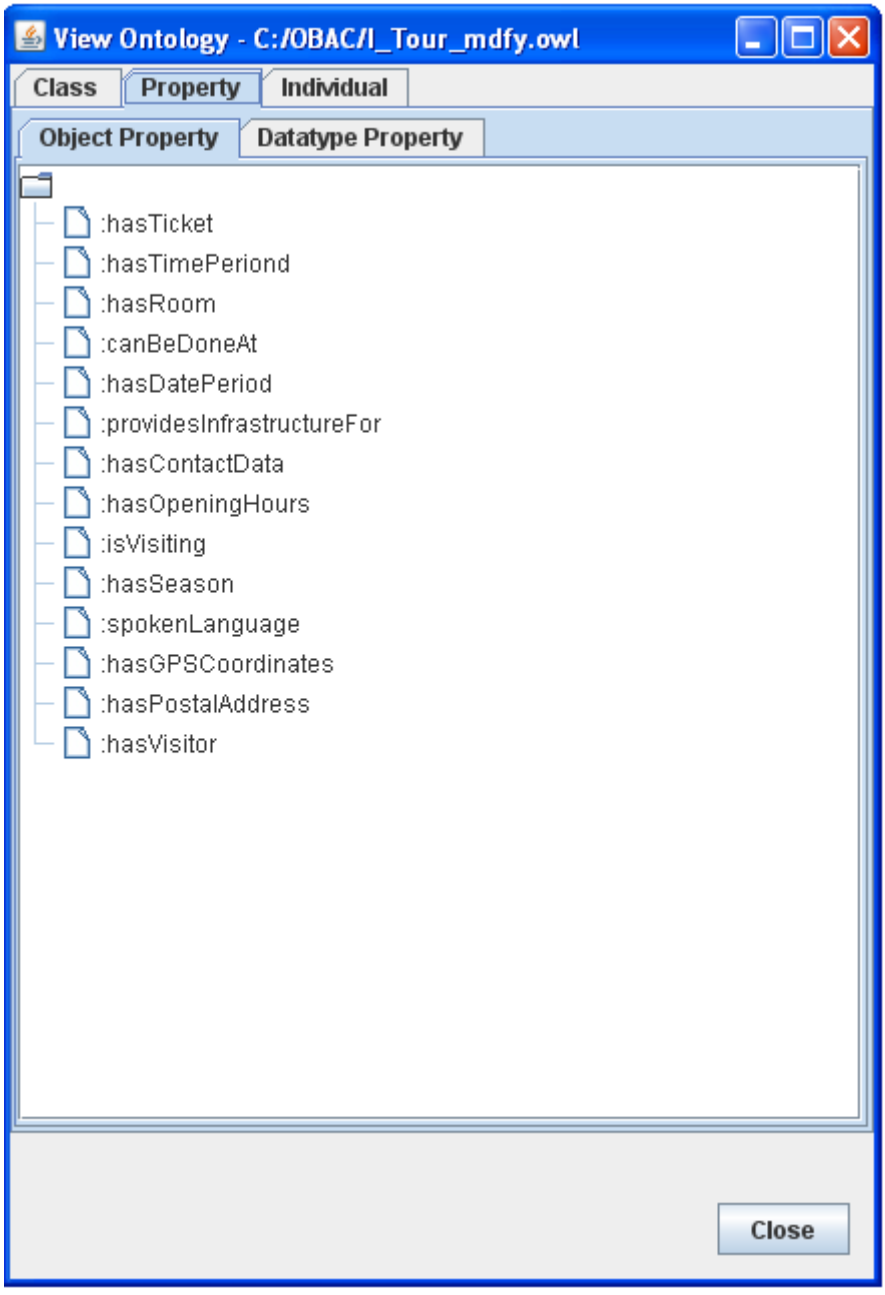
Şekil 7.53 Görüntülenecek ontolojinin belirtilmesi.

Bu bölümde örnek olarak seçilen `I_Tour_mdfy.owl` dosyasının sınıf sıradüzenseli görüntüsü Şekil 7.54'de yer almaktadır. Ontolojinin özellik sıradüzenseli; nesne (*object*) ve veri türü (*datatype*) olmak üzere iki şekilde görüntülenmektedir. Aynı ontolojinin nesne özellik sıradüzenseli Şekil 7.55'de, veri türü özellik sıradüzenseli ise Şekil 7.56'da gösterilmiştir.

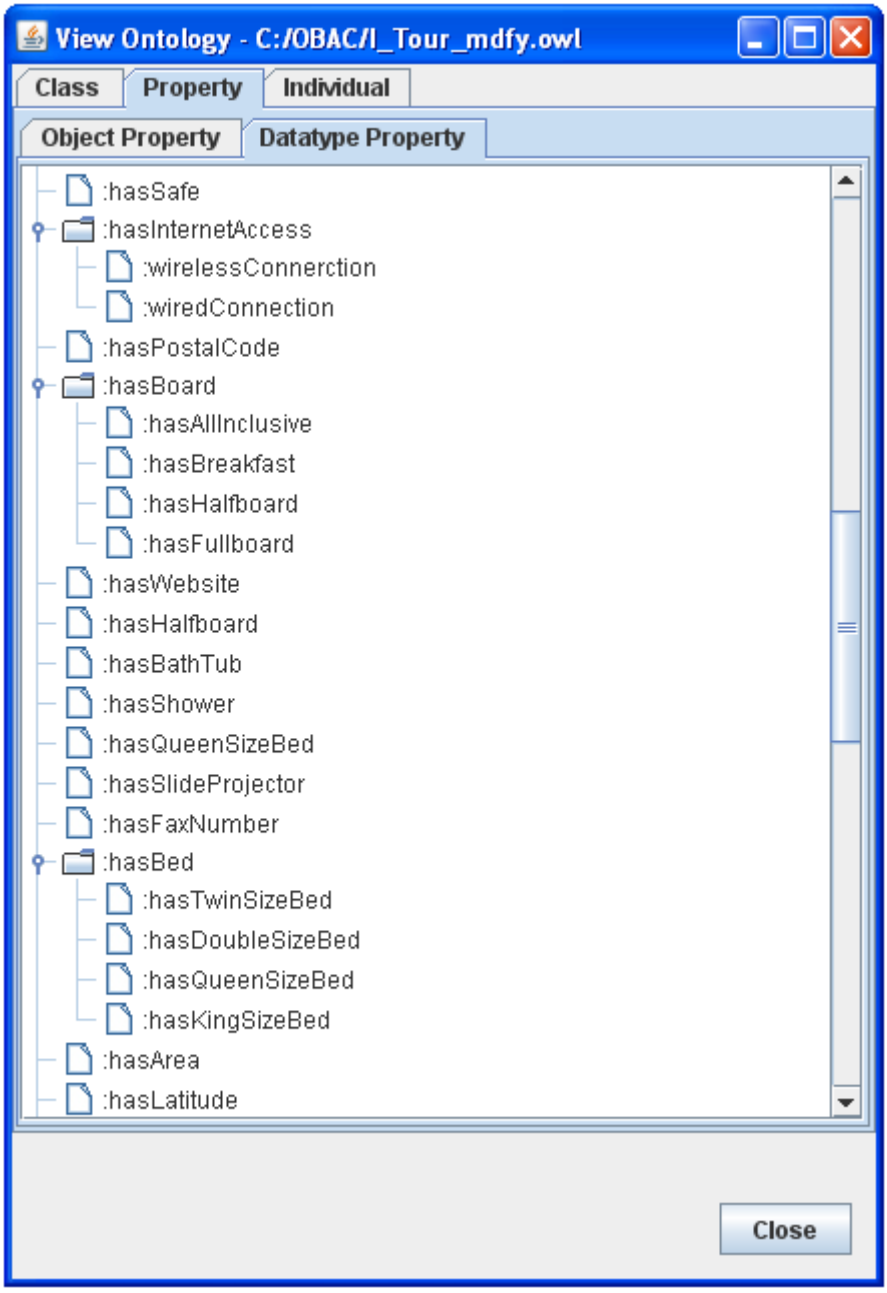
Ontolojinin örnek sıradüzenseli görüntüsü Şekil 7.57'de yer almaktadır. Burada her bir sınıfa ilişkin örnekler o sınıfın altında belirtilmiştir.



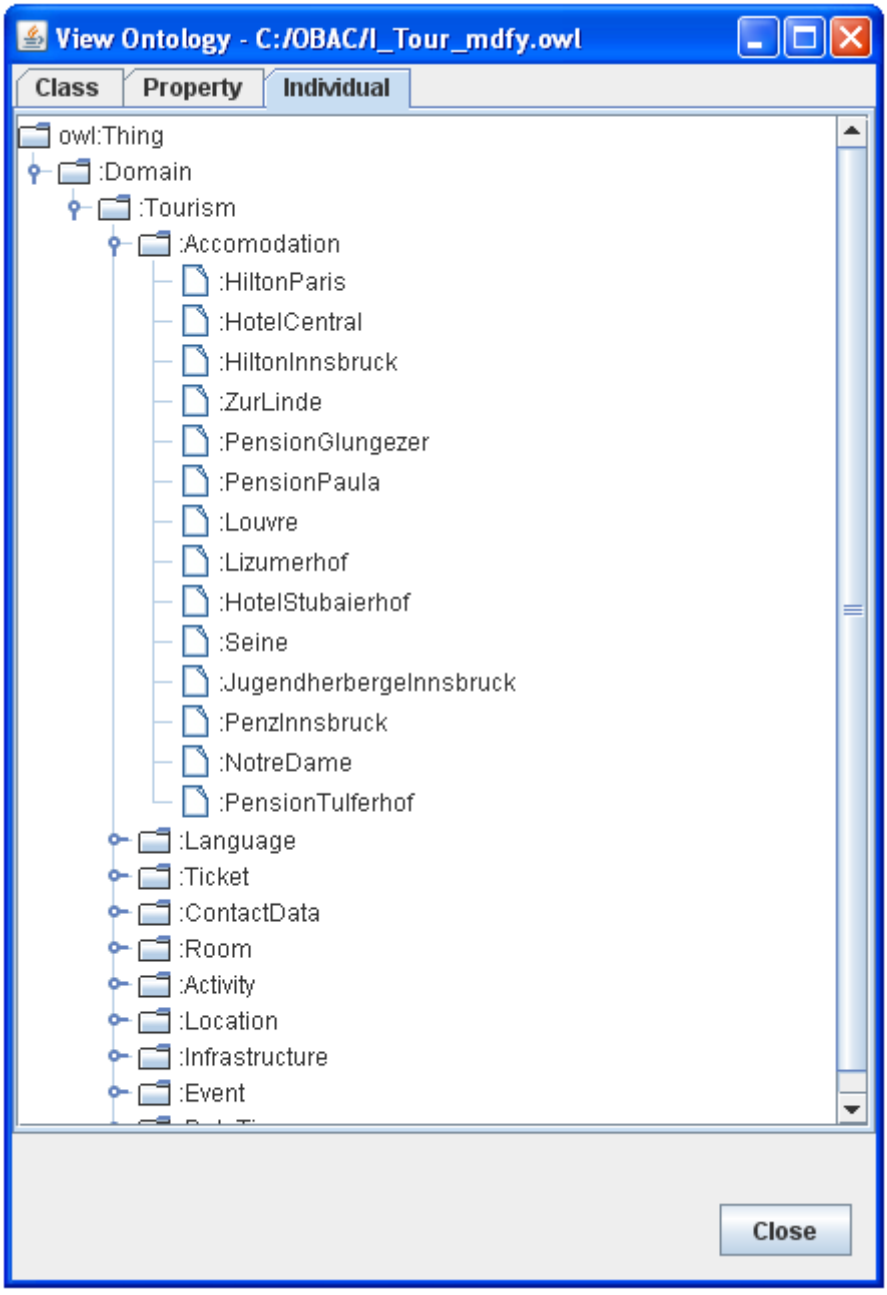
Şekil 7.54 Ontolojinin sınıf sıradüzenseli.



Şekil 7.55 Ontolojinin nesne özellik sıradüzenseli.



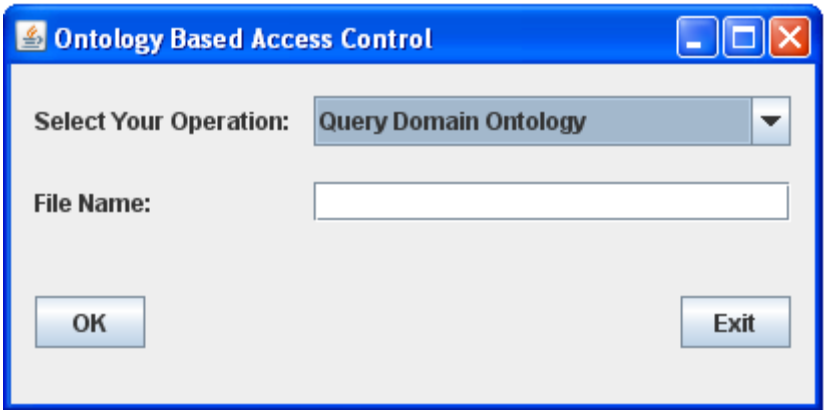
Şekil 7.56 Ontolojinin veri türü özellik sıradüzenseli.



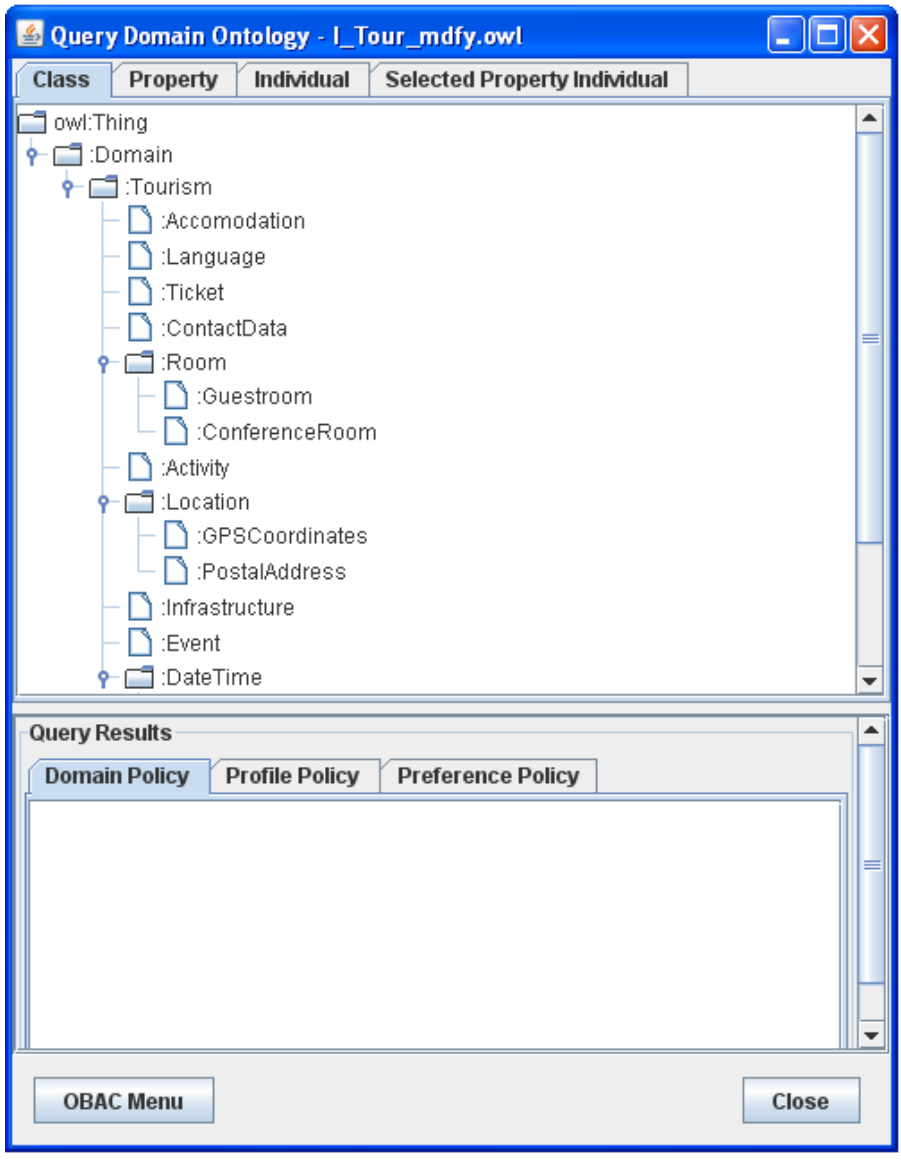
Şekil 7.57 Ontolojinin örnek sıradüzenseli.

7.4.5 Etki alanı ontolojisinin sorgulanması

Etki alanı ontolojisinin sorgulanmasında sistem kendiliğinden `I_Tour_mdfy.owl` dosyasını açacağı için kullanıcı herhangi bir dosya adı girmemektedir. Şekil 7.58'de görüleceği gibi kullanıcı, etki alanı ontolojisinin sorgulanması (*query domain ontology*) işlemini seçtiktikten sonra, Şekil 7.59'da yer alan `I_Tour_mdfy.owl` etki alanı ontolojisinin sırasıyla sınıf, özellik, örnek ve seçilmiş özelliğin örnek sıradüzensellerini görüntülemektedir.



Şekil 7.58 Etki alanı ontolojisinin sorgulanması işlemi.



Şekil 7.59 Etki alanı ontolojisinin sorgulanması.

Etki alanı ontolojisinin sınıf, özellik ve örnek sıradüzenselleri ontoloji görüntüleme kısmında anlatılanlar ile aynıdır. Ancak, etki alanı ontolojisinin sorgulanmasında, ontoloji görüntüleme kısmından farklı olarak kullanıcı tarafından seçilmiş sınıfın, örneğin ya da seçilmiş bir özellik ile ilgili politika ontolojileri içerisinde sorgulamalar gerçekleştirilmektedir. Ayrıca, Seçilmiş Özellik Örneği – Selected Property Individual sekmesinde seçilmiş bir özelliği kullanan örnek sıradüzensellerinin görüntülenmesi yer almaktadır. Bu sekmede, bütün örneklerin listelenmesi yerine sadece özellik sıradüzenselinden seçilmiş olan herhangi bir özelliğin kullanıldığı örnekler listelenmektedir.

Örneğin; `hasSnackBar` veri türü özelliği seçildiğinde bu özelliğin "`true`" değerinin olduğu oteller listelenmektedir. Şekil 7.60'da listelenen 4 oteli (`HiltonParis, Louvre, Seine, NotreDame`) göstermektedir.

Etki alanı ontolojisinin sorgulanması penceresinin alt kısmında yer alan Sorgu Sonuçları – Query Results kısmında ise seçilen sınıf, özellik ve örnek ile ilgili sorgu sonuçları listelenmektedir. Daha iyi bir kişiselleştirmenin sağlanabilmesi için sorgular politikalar üzerinden gerçekleştirilmektedir. Böylelikle kullanıcı için sorgu sonuçlarıda daraltılmış olmaktadır. Sorgu sonuçları üç bölümden oluşmaktadır:

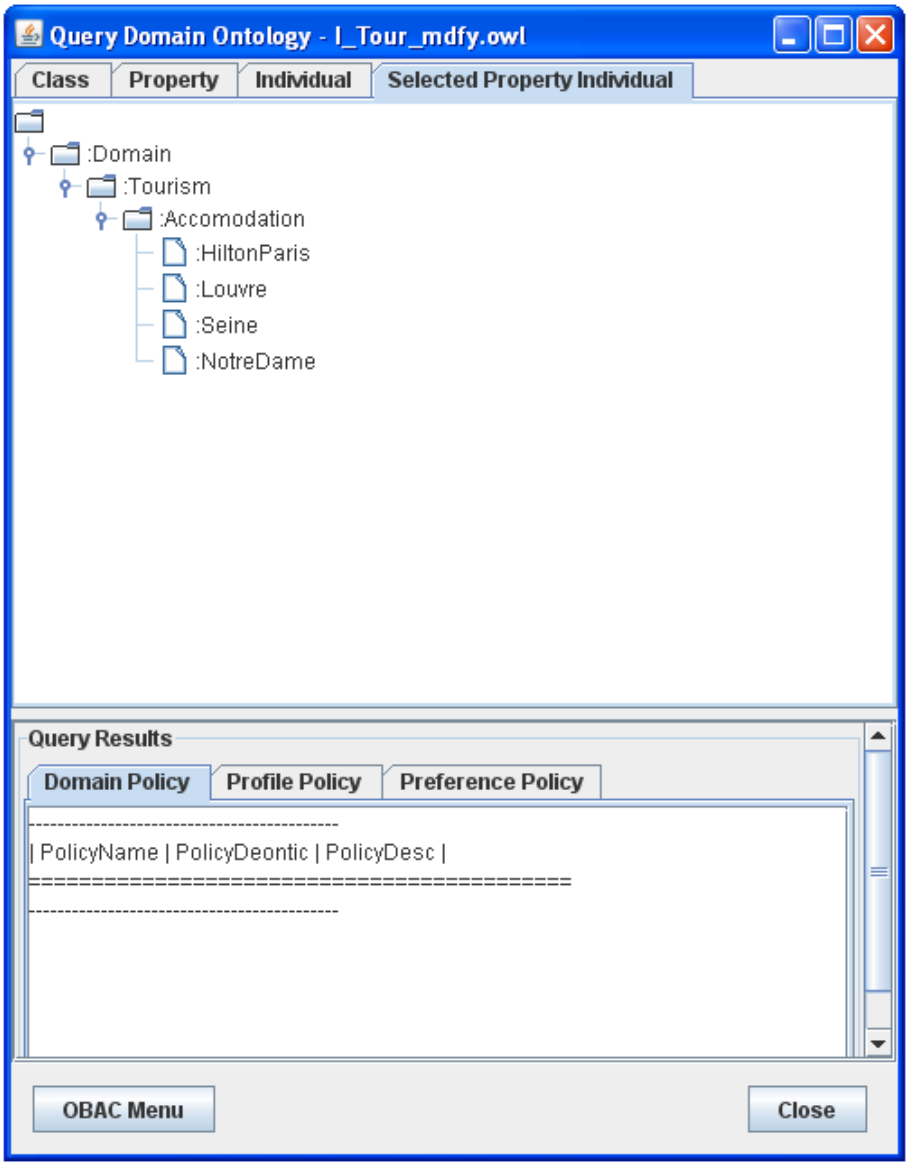
- Etki Alanı Politikası – Domain Policy
- Profil Politikası – Profile Policy
- Seçenek Politikası – Preference Policy

Seçilen sınıf, özellik ya da örnek ile ilgili sorgu işlemleri her 3 politika ontolojisi içerisinde çalıştırıldığından her bir ontolojiden gelen sonuçlar ilgili ontolojinin sekmesinde listelenmektedir.

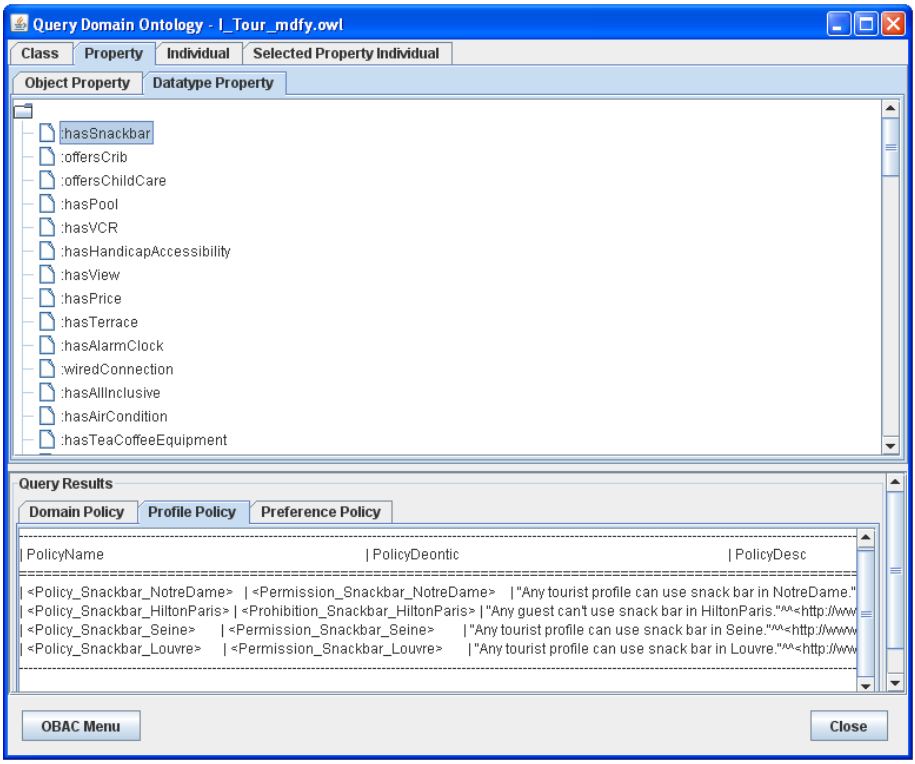
Yukarıda verilen "`hasSnackBar=true`" örneği ile ilgili sorgu sonuçları profil ve seçenek politika ontolojilerinden dönmektedir. Listenen sonuçlar Şekil 7.61'de ve Şekil 7.62'de gösterilmiştir.

Profil politika ontolojisinden gelen sorgu sonuçlarını gösteren Şekil 7.61'de 2 yasak 2 izin olmak üzere 4 sonuç yer almaktadır. Bu sonuçların elde edildiği profil politika ontolojisinde çalıştırılan sorgu aşağıda yer almaktadır. Burada politika adı, politika deontik nesnesinin türü ve politika açıklaması listelenmektedir.

```
final Query queryString = QueryFactory.create(
"PREFIX dc:
<http://efe.ege.edu.tr/~ozgucan/PhD/Ontology/I_Tour_mdfy.ow
l#> " +
"PREFIX tourism:
<http://efe.ege.edu.tr/~odo/Ontology/I_Tour_mdfy.owl#>"+
"PREFIX policy:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiPolicy.o
wl#>"+
"PREFIX action:
<http://www.csee.umbc.edu/~lkagal1/rei/ontologies/ReiAction
.owl#>"+
"PREFIX constraint:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiConstrai
nt.owl#>"+
"SELECT ?PolicyName ?PolicyDeontic ?PolicyDesc WHERE {
?PolicyName policy:action ?xy."+
"?xy action:precondition ?z ." +
"?z constraint:object tourism:"+selectedNode+" ."+
"?PolicyName policy:grants ?w . "+
"?w policy:deontic ?PolicyDeontic ."+
"?PolicyDeontic policy:desc ?PolicyDesc }");
```

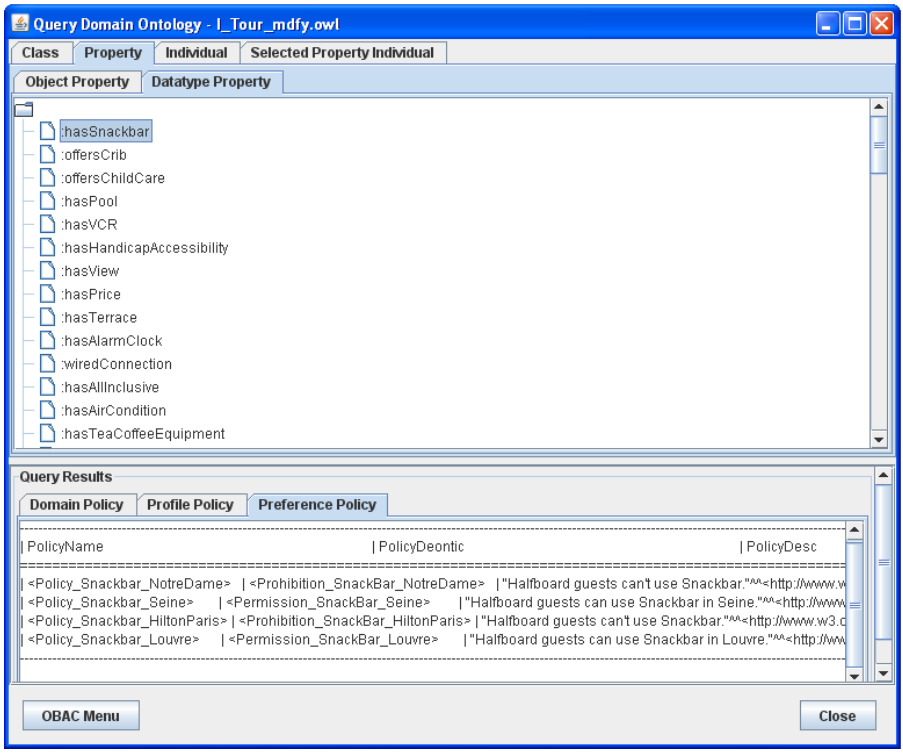


Şekil 7.60 Seçilmiş bir özelliğin örneklerinin listelenmesi.



Şekil 7.61 `hasSnackBar` özelliği ile ilgili profil politikasından gelen sorgu sonuçları.

Seçenek politika ontolojisinden gelen sorgu sonuçlarını gösteren Şekil 7.62'de de profil politika ontolojisinde olduğu gibi 2 yasak 2 izin olmak üzere 4 sonuç yer almaktadır. Bu sonuçların elde edildiği seçenek politika ontolojisinde çalıştırılan sorgu da profil politika ontolojisi için çalıştırılan sorgu ile aynıdır.



Şekil 7.62 `hasSnackBar` özelliği ile ilgili seçenek politikasından gelen sorgu sonuçları.

Seçilen özelliğin etki alanı politika ontolojisinde sorgulanmasında çalıştırılan sorgu aşağıda yer almaktadır. Bu sorgu sonucunda politika adı, politika deontik nesnesinin türü ve politika açıklaması listelenmektedir.

```
final Query queryStringDomain = QueryFactory.create(
"PREFIX dc:
<http://efe.ege.edu.tr/~ozgucan/PhD/Ontology/I_Tour_mdfy.ow
l#> " +
"PREFIX tourism:
<http://e-tourism.deri.at/ont/e-tourism.owl#>"+
```

```
"PREFIX policy:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiPolicy.o
wl#>"+
"PREFIX action:
<http://www.csee.umbc.edu/~lkagal1/rei/ontologies/ReiAction
.owl#>"+
"PREFIX constraint:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiConstrai
nt.owl#>"+
"SELECT ?PolicyName ?PolicyDeontic ?PolicyDesc WHERE {
?PolicyName policy:action ?xy."+
"?xy action:precondition tourism:" +selectedNode+" ."+
"?PolicyName policy:grants ?w . "+
"?w policy:deontic ?PolicyDeontic ."+
"?PolicyDeontic policy:desc ?PolicyDesc }");
```

Jena kullanılarak gerçekleştirilen bu sorguların çalıştırılması ise şu şekilde olmaktadır:

```
QueryExecution qexec = QueryExecutionFactory.create
(queryStringDomain, tourismPolicyModel) ;
ResultSet results = qexec.execSelect() ;
ResultSetFormatter fmt=new ResultSetFormatter();
```

Seçilen herhangi bir sınıf ve örneğin etki alanı, profil ve seçenek politika ontolojilerinde gerçekleştirilen sorguları ise aşağıdaki gibidir.

Etki alanı politika ontolojisi içerisinde;

```
final Query queryString = QueryFactory.create(
"PREFIX dc:
<http://efe.ege.edu.tr/~ozgucan/PhD/Ontology/I_Tour_mdfy.ow
l#> " +
"PREFIX policy:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiPolicy.o
wl#>"+
"PREFIX action:
<http://www.csee.umbc.edu/~lkagal1/rei/ontologies/ReiAction
.owl#>"+
"SELECT ?PolicyName ?PolicyDeontic WHERE {
```

```
?PolicyName policy:action ?xy."+
"?xy action:location dc"+selectedNode +" ." +
"?PolicyName policy:grants ?z . "+
"?z policy:deontic ?PolicyDeontic }");
```

Profil politika ontolojisi içerisinde;

```
final Query queryString = QueryFactory.create(
"PREFIX dc:
<http://efe.ege.edu.tr/~ozgucan/PhD/Ontology/I_Tour_mdfy.ow
l#> " +
"PREFIX policy:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiPolicy.o
wl#>"+
"PREFIX action:
<http://www.csee.umbc.edu/~lkagal1/rei/ontologies/ReiAction
.owl#>"+
"PREFIX pvd_domain:
<http://efe.ege.edu.tr/~odo/Ontology/PVD_Domain.owl#>"+
"SELECT ?PolicyName ?PolicyDeontic WHERE {
?PolicyName policy:action ?xy."+
"?xy action:location pvd_domain"+selectedNode +" ." +
"?PolicyName policy:grants ?z . "+
"?z policy:deontic ?PolicyDeontic }");
```

Seçenek politika ontolojisi içerisinde;

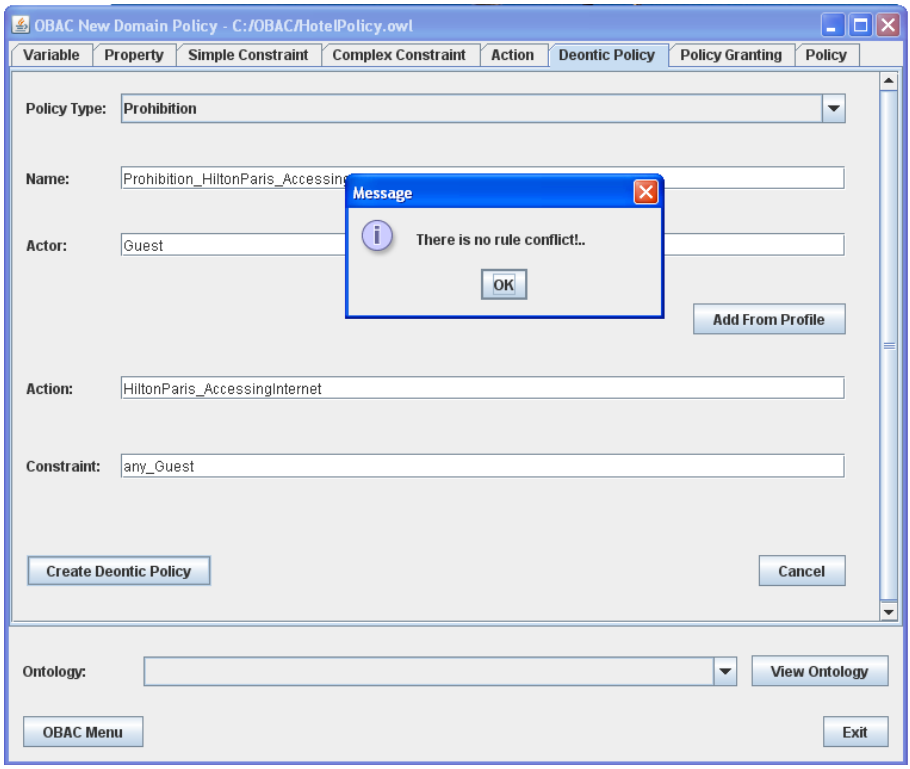
```
final Query queryString = QueryFactory.create(
"PREFIX dc:
<http://efe.ege.edu.tr/~ozgucan/PhD/Ontology/I_Tour_mdfy.ow
l#> " +
"PREFIX policy:
<http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiPolicy.o
wl#>"+
"PREFIX action:
<http://www.csee.umbc.edu/~lkagal1/rei/ontologies/ReiAction
.owl#>"+
"PREFIX pvd_domain:
<http://efe.ege.edu.tr/~odo/Ontology/PVD_Domain.owl#>"+
"SELECT ?PolicyName ?PolicyDeontic WHERE {
?PolicyName policy:action ?xy."+
"?xy action:location pvd_domain"+selectedNode +" ." +
"?PolicyName policy:grants ?z . "+
```

```
"?z policy:deontic ?PolicyDeontic }");
```

7.4.6 Çelişkilerin Çözümü

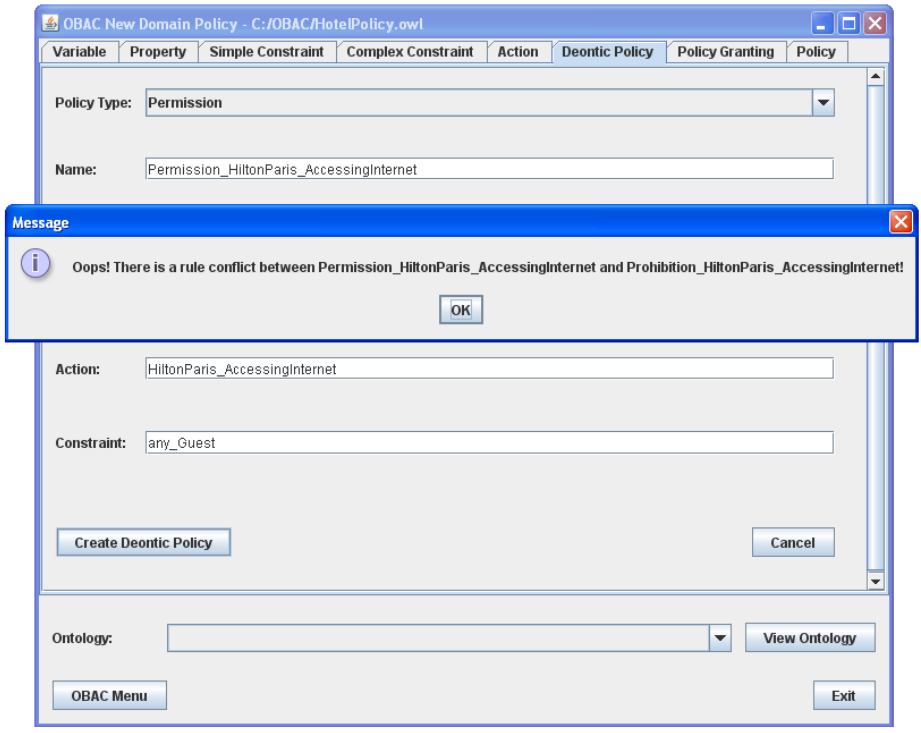
Politika ontolojisi oluşturulurken ortaya çıkabilecek çelişkilerin saptanması ve çözümü program içerisinde iki yerde yapılmaktadır. Bunların ilki deontik politika tanımı yapılırken ikincisi ise politika tanımının yapılması sırasındadır. Yukarıda bahsedilen ilk çelişkinin çözümü kuralların çelişmesi ile ikinci çelişkinin çözümü ise politikaların çelişmesi ile çözülmektedir.

Örneğin; Hilton Paris otelinde internet erişimi yasağı ile ilgili bir politika kuralı yaratılırken herhangi bir kural çelişmesinin meydana gelip gelmediği denetlenmektedir. Bu örnek için herhangi bir kural çelişmesi olmadığı Şekil 7.63'de görülmektedir. Daha sonra aynı otel için internet erişimi izin kuralı yaratılmaktadır. Bu durumda bir kural çelişmesi oluşmaktadır. Şekil 7.64'de kural çelişmesi iletisi yer almaktadır.

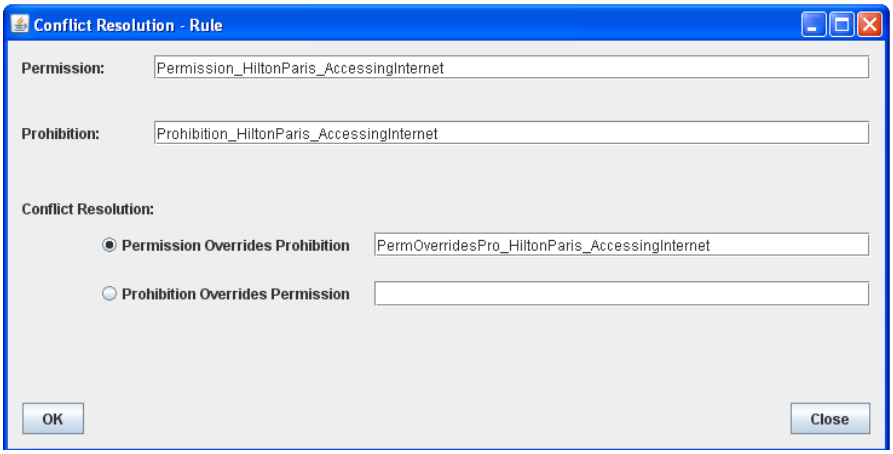


Şekil 7.63 Yasak kuralı tanımlanırken kural çelişmesinin denetlenmesi.

Oluşan kural çelişmesinin çözümü için Şekil 7.65'de görülen ekran görüntüsü etkin olmaktadır. Kullanıcı, iznin yasağa ya da yasağın izne olan üstünlüğünü burada tanımlamaktadır. Bu örnekte izin yasağın üzerine yazmaktadır.

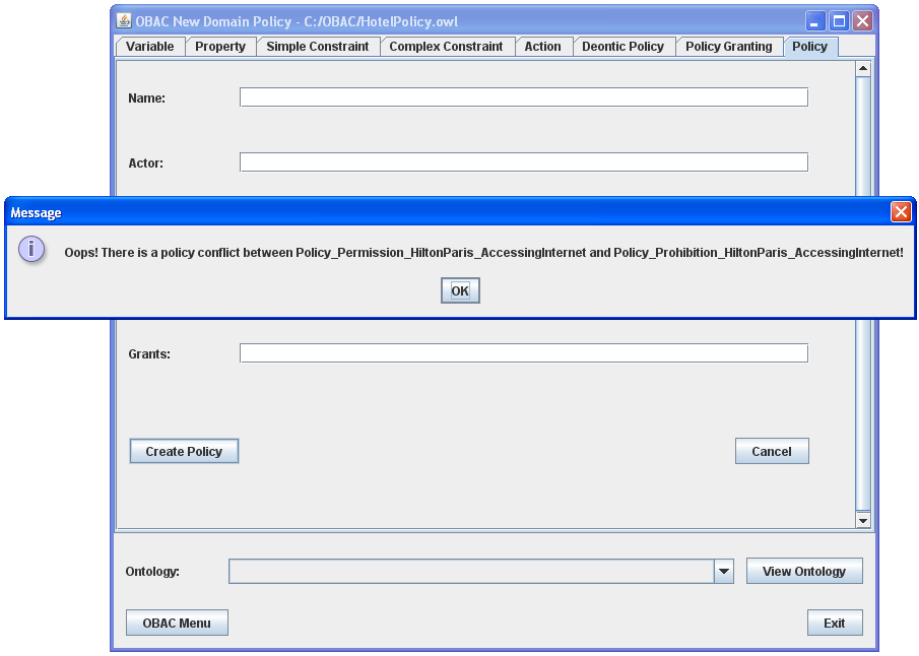


Şekil 7.64 İzin kuralı tanımlanırken kural çelişmesinin oluşması.

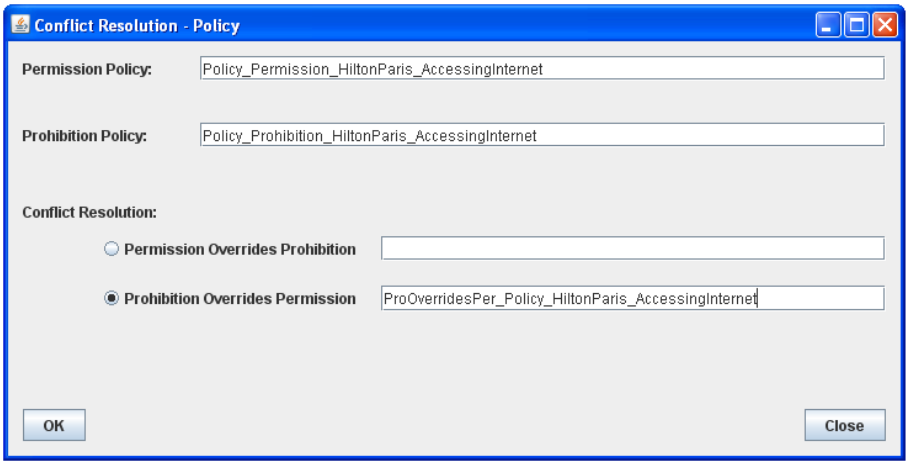


Şekil 7.65 Kural çelişmesi çözümü.

Yukarıda verilen örneğe devam ettiğimizde Hilton Paris otelinde internet erişimi için biri yasak biri izin iki politika tanımlandığında bir politika çelişmesi ortaya çıkacaktır. Şekil 7.66'da politika çelişmesinin ortaya çıkması yer almaktadır. Şekil 7.67'de ise politika çelişmesinin çözümü görülmektedir. Kullanıcı, yasak politikasının izin politikasından daha büyük bir önceliğinin olduğunu tanımlamaktadır.



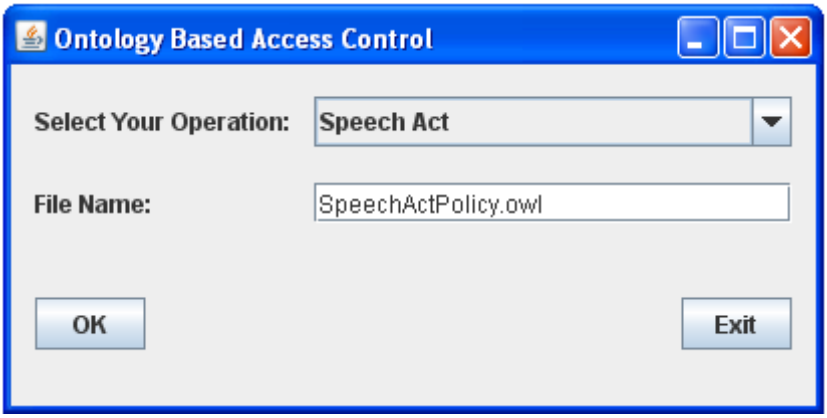
Şekil 7.66 Politika tanımlanırken politika çelişmesinin oluşması.



Şekil 7.67 Politika çelişmesinin çözümü.

7.4.7 Konuşma Edimleri

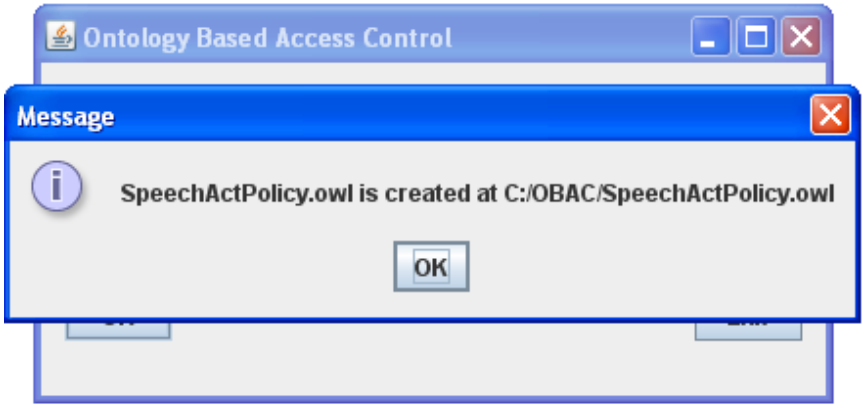
Güvenlik denetimini sağlamak amacı ile kullanılan konuşma edimlerinin oluşturulması için kullanıcı öncelikle Şekil 7.68'de görüldüğü gibi dosya adını belirtmektedir. Şekil 7.69'da dosyanın oluşturulduğu iletisi görülmektedir.



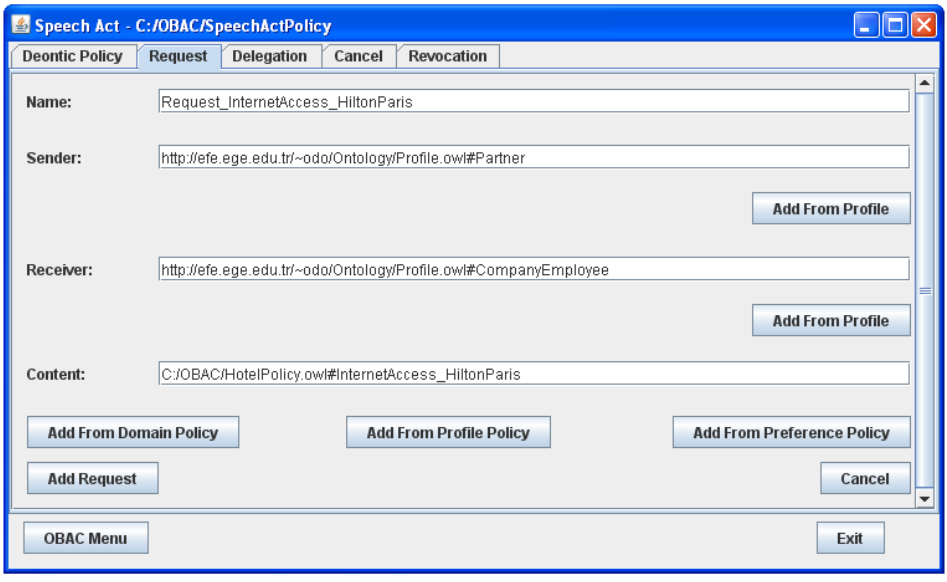
Şekil 7.68 Konuşma edimi dosya adının belirtilmesi.

Şekil 7.70'de yer alan konuşma edimi penceresi 5 sekmeden oluşmaktadır:

- İstek (Request)
- Yetki aktarımı (Delegation)
- İptal (Cancel)
- Yetkinin geri alımı (Revocation)
- Deontik Politika (Deontic Policy)



Şekil 7.69 Konuşma edimi dosyasının oluşturulması.



Şekil 7.70 Konuşma edimi penceresi.

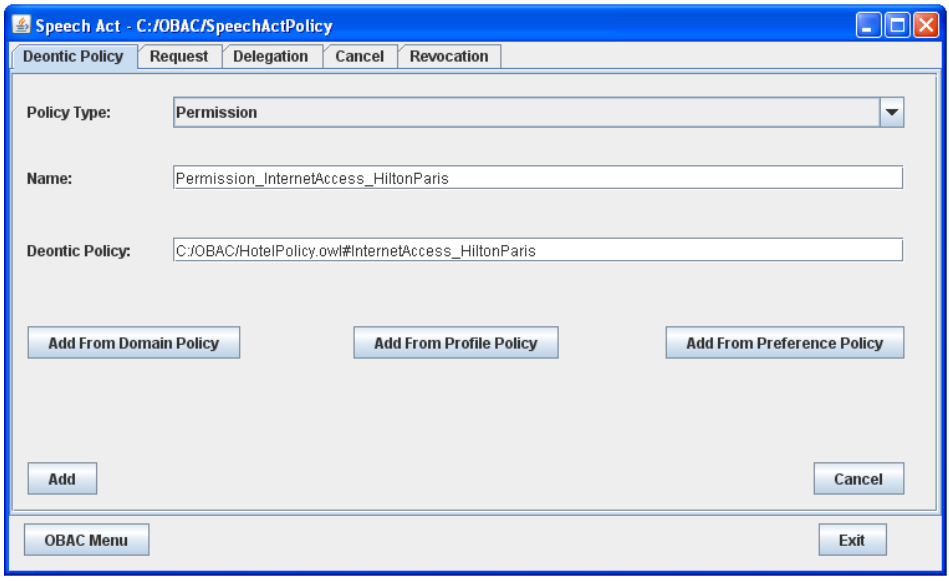
İstek, Yetki aktarımı, İptal ve Yetkinin geri alımı sekmeleri sırası

Şekil 7.70'de de görülen aşağıdaki dört alandan oluşmaktadır:

- ***Ad – Name:*** Oluşturulan konuşma ediminin adıdır.
- ***Gönderici – Sender:*** Konuşma edimi ile ilgili istekte bulununan varlıktır.
- ***Alıcı – Receiver:*** Konuşma edimi ile ilgi isteği alan varlıktır.
- ***İçerik – Content:*** Konuşma ediminin ilgili olduğu içeriği belirtmektedir.

Deontik politika sekmesi, oluşturulacak deontik politikanın türü, adı ve etki alanı, profil, seçenek ya da başka bir politika ontolojisinden alınan deontik politika alanlarından oluşmaktadır. Şekil 7.71'de deontik politika sekmesi yer almaktadır.

Konuşma edimi penceresinde OBAC Menü'süne geri dönmeyi sağlayan OBAC Menü – OBAC Menu ve programdan çıkış için Çıkış – Exit düğmeleri yer almaktadır.



Şekil 7.71 Deontik politika sekmesi.

8. SONUÇLAR

Bu tez çalışmasında Anlamsal Web uygulamalarında güvenliğin ve kişiselleştirmenin sağlanabilmesi için Ontoloji Tabanlı Erişim Denetimi geliştirilmesine yönelik olarak çeşitli çalışmalar gerçekleştirilmiştir. Bu çalışmalarda politikaların kullanımından yararlanılmıştır. Geleneksel erişim denetim düzeneklerinde politikalar kaynağa erişimin denetiminde kullanılırken, bu tezde politikalar erişim denetiminin yanı sıra erişim denetiminde kişiselleştirmeyi gerçekleştirmek için seçeneklerin ve profillerin yönetiminde de kullanılmıştır. Gerçekleştirilen çalışmalar aşağıdaki başlıklarda toplanabilir:

- ✓ **Politika Ontolojilerinin Geliştirilmesi:** Rei politika dili kullanılarak üç çeşit politika geliştirilmiştir. Bunlar: *etki alanı*, *profil* ve *seçenek* tabanlı politikalardır.

- ✓ **Uygulama Geliştirimi:** Java programlama dili ve Jena Anlamsal Web çatısı kullanılarak ontolojilerin Java ortamında yaratılması, düzenlenmesi, silinmesi ve görüntülenmesi ile ilgili bir politika gerçekleştirim aracı geliştirilmiştir.

- ✓ **Politika Çelişkilerinin Çözümü:** Politikalar oluşturulurken ortaya çıkan kural ve politika çelişkileri saptanıp çözülmektedir.

- ✓ **Sorguların Gerçekleştirimi:** Geliştirilen politika aracı kullanılarak, ontoloji örnekleri üzerinden çeşitli SPARQL sorguları çalıştırılmış ve kişiselleştirilmiş sorgu sonuçlarına ulaşılması sağlanmıştır.

Bu çalışmanın ardından yapılabilecek diğer bazı çalışmalar aşağıda belirtilmiştir:

- ➢ Bu tez çalışmasında geliştirilen durum çalışması, turizm etki alanını kapsamaktadır. Bundan sonra yapılacak çalışmalar için uygulama alanı değiştirilebilir veya başka etki alanları da durum çalışmasına eklenerek uygulama alanı genişletilebilir.

- ➢ Ontolojilere erişimin denetimi gerçekleştirilmelidir. Erişim düzeneği ile ilgili bilgilerin modellenmesi için ontoloji tabanlı politikaların yaratılmasının yanı sıra ontolojilere erişimin denetlenmesi de bu gelecek çalışmanın hedefleri arasında yer almaktadır. Ontolojileri kullanarak erişim politikaların tanımlanmasının yanı sıra ontolojilerin kendi güvenliğinin de sağlanması gereklidir. Ontolojilerin bazı bölümleri gizli bazı bölümleri ise açık olabilir. Ontolojiler sürekli geliştiğinden ontolojilere erişim değişiklik gösterebilir. Ontolojinin farklı bölümlerine farklı kullanıcıların erişim yetkisi olabilir. Güvenli bilgi bütünleşmesi için ontolojilerin nasıl kullanılacakları önem

kazanmaktadır.

- Doğal Dil İşleme (Natural Language Processing - NLP) yöntemleri dizgeye eklenerek kullanıcıların doğal dil cümleleri ile politika sonuçlarına ulaşmaları gerçekleştirilebilir.

- Geliştirilen araç örgü ortamına taşınıp, örgü üzerinden çalışacak duruma dönüştürülebilir. Böylelikle dizgenin gerçek yaşam uygulamaları ile bütünleştirilip güncel bir biçimde kullanılması sağlanabilir.

- İndirilebilir bir Rei politika motoru bulunmadığından OBAC dizgesi açık kaynak kod durumuna getirilip internete konulacaktır.

KAYNAKLAR DİZİNİ

Abou El Kalam, A., El Baida, R., Balbiani,P., Benferhat, S., Cuppens, F., Deswarte,Y., Miège, A., Saurel, C. and Trouessin,G., 2003, Organization Based Access Control, IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Come, Italy, June 4-6.

Anderson, A. H., 2004, An Introduction to the Web Services Policy Language (WSPL), 5th IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights, New York, 7-9 June 2004.

Antoniou, G. and van Harmelen, F., 2004, A Semantic Web Primer, The MIT Press, ISBN 0-262-01210-3.

Benantar, M., 2006, Access Control Systems Security, Identity Management and Trust Models, Springer Science Business Media, ISBN: 978-0-387-00445-7.

Bonatti, P. A. and Olmedilla, D., 2005, Policy Language Specification, Technical report, Working Group I2, EU NoE REWERSE.

Bonatti, P. and Samarati, P., 2000, Regulating Service Access and Information Release on the Web. In Conference on Computer and Communications Security (CCS'00), Athens, November 2000.

KAYNAKLAR DİZİNİ (devam)

Bursa, O., Ünalır, M. O., 2008, Anlamsal Web Portallarda Profil Yönetimi, Ulusal Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu 08 (YKS'08), İstanbul, Türkiye.

Chen T.-Y., 2008, Knowledge sharing in virtual enterprises via an ontology-based access control approach. Computers in Industry, 59(5), 502–519.

Clemente, F. J. G., Perez, G. M., Blaya, J. A. B. and Skarmeta, A. F. G., 2005, Representing Security Policies in Web Information Systems, Workshop on Policy Management for the Web, 14th International WorldWideWeb Conference, WWW 2005, Chiba (Japan). Conference proceedings, 61-66pp.

Cranor, L., Langheinrich, M., Marchiori, M., 2002, A P3P Preference Exchange Language 1.0 (APPEL1.0), W3C Working Draft, April 15.

Cuppens, F., Miège, A., 2003, Modelling Contexts in the Or-BAC Model, 19th Annual Computer Security Applications Conference.

Daconta, M. C., Obrst L. J., Smith, K. T., 2003, The Semantic Web:A Guide to the Future of XML, Web Services, and Knowledge Management, Wiley Publishing, ISBN 0-471-43257-1.

KAYNAKLAR DİZİNİ (devam)

Damiani, E., De Capitani di Vimercati, S., Fugazza,C. and Samarati, P., 2004, Extending Policy Languages to the Semantic Web, Web Engineering 4th International Conference, ICWE 2004, Munich, Germany, July 26–30, 330–343pp.

Damianou, N., Dulay, N., Lupu, E., Sloman, M., 1995, The Ponder Policy Specifiication Language, Workshop on Policies for Distributed Systems and Networks, Springer-Verlag LNCS, 18-39pp.

De Coi, J. L., Olmedilla, D., Bonatti, P.A. and Sauro, L., 2008, Protune: A framework for semantic web policies. In Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008, volume 401 of CEUR Workshop Proceedings. CEUR-WS.org.

De Coi, J. L. and Olmedilla, D., 2008, A Review of Trust Management, Security and Privacy Policy Languages, In International Conference on Security and Cryptography (SECRYPT 2008), INSTICC Press.

Dulay, N., Lupu, E., Sloman, M., Damianou, N., 2001, A Policy Deployment Model for the Ponder Language, Proc. IEEE/IFIP International Symposium on Integrated Network Management (IM'2001).

KAYNAKLAR DİZİNİ (devam)

Duma, C., Herzog, A., Shahmehri, N., 2007, Privacy in the SemanticWeb: What Policy Languages Have to Offer, Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07), 109-118pp.

Dzbor, M., Motta, E., 2008, Engineering and Customizing Ontologies, Ontology Management, Semantic Web, Semantic Web Services, and Business Applications, 25-57pp.

Erdur, R. C., 2001, YazılımEtmeni Teknolojisinin İnternet Tabanlı Yazılım Yeniden Kullanımına Uygulanması, Doktora Tezi.

Ferraiolo, D., and Kuhn, D. R., 1992, Role-Based Access Controls, Proceedings of the NIST-NSA National (USA) Computer Security Conference, 554–563pp.

Ferraiolo, D. F., Kuhn, D. R., Chandramouli, R., 2007, Role Based Access Control, Artech House Publishers, Second Edition, ISBN 13: 978-1-59693-113-8.

Finin, T. et al., 2008, ROWLBAC - Representing Role Based Access Control in OWL, Proceedings of the 13th Symposium on Access Control Models and Technologies.

KAYNAKLAR DİZİNİ (devam)

Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A., 2007, User Profiles for Personalized Information Access, The Adaptive Web 2007, 54-89pp.

Godik, S., Moses, T., eds., 2003, OASIS eXtensible Access Control Markup Language (XACML) Version 1.1, OASIS Committee Specification, http://www.oasis-open.org/committees/download.php/ 4103/ cs -xacml-specification-1.1.doc, July 24.

Gruber, T.R., 1993, Toward principles for the Design of Ontologies Used for Knowledge Sharing, International Journal of Human-Computer Studies, Volume 43 , Issue 5-6, 907-928.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., 2004, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C, http://www.w3.org/Submission/SWRL/ .

Huang, D., 2005, Semantic Policy-based Security Framework for Business Processes, 4th International Semantic Web Conference, Galway, Ireland, November 7.

Jrad, Z., Aufaure, M.A., 2007, Personalized Interfaces for a Semantic Web Portal. Tourism Information Search, In KES 2007/WIRN 2007, Part III, LNAI 4694, 695-702pp.

KAYNAKLAR DİZİNİ (devam)

Kagal, L., 2002, Rei:A Policy Language for the Me-Centric Project, TechReport, HP Labs.

Kagal, L., Finin, T., Joshi, A., 2003, A Policy Based Approach to Security for the Semantic Web, 2nd International Semantic Web Conference (ISWC 2003), 402-418pp.

Kagal, L., Finin, T., Paolucci, M., Srinivasen, N., Sycara, K., Denker, G., 2004, Authorization and Privacy for Semantic Web Services, IEEE Intelligent Systems, vol. 19, no. 4, 50-56 pp. , July/Aug. 2004, doi:10.1109/MIS.2004.23.

Kagal, L. and Finin, T., 2005, Rein : Where Policies Meet Rules in the Semantic Web, Technical report, MIT.

Katifori, A., Golemati, M., Vassilakis, C., Lepouras, G., Halatsis, C., 2007, Creating an Ontology for the User Profile: Method and Applications, In the proceedings of the First IEEE International Conference on Research Challenges in Information Science (RCIS), Morocco, 407-412pp.

Lassila, O., 2005, Applying Semantic Web in Mobile and Ubiquitous Computing: Will Policy-Awareness Help?, 4th International Semantic Web Conference, Galway, Ireland, November 7.

KAYNAKLAR DİZİNİ (devam)

Lupu, E. and Sloman M., 1999, Conflicts in Policy-based Distributed Systems Management, IEEE Transactions on Software Engineering, 25:852-869pp.

Priebe, T., Dobmeier, W., Kamprath, N., 2006, Supporting Attributed-based Access Control with Otologies, Proc. of the First International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria, 465–472pp.

Priebe, T., Dobmeier, W., Schläger, C., Kamprath, N., 2007, Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies. Journal Of Software (JSW), ISSN: 1796-217X, Vol. 2, Iss. 1, 27–38pp.

Rich, E., 1999, Users are individuals: Individualizing User Models, Int. J. Hum. Comput. Stud., 51(2): 323-338.

Quinn, K., O'Sullivan, D., Wade, V., 2004, Policy Driven Composition of Trustworthy Web Services, 4th annual Conference on Information Technology and Telecommunications, IT&T, Limerick, Ireland, October 20-21.

KAYNAKLAR DİZİNİ (devam)

Studer, R., Benjamins, V. R., Fensel, D., 1998, Knowledge Engineering: Principles and Methods, Data Knowl. Eng. (1998) 25(1-2): 161-197pp.

Su, L., Chadwick, D. W., Basden, A., Cunningham, J. A., 2005, Automated Decomposition of Access Control Policies, Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05), Stockholm, Sweden, 6–8 June, 3–13pp.

Sun, Y., Pan, P., Leung, H., and Shi, B., 2007, Ontology Based Hybrid Access Control for Automatic Interoperation, 4th International Conference, ATC 2007, Hong Kong, China, July 11-13, 323-332pp.

Tapucu, D., Can, Ö., Bursa, O., Ünalır, M. O., 2008, Metamodeling Approach to Preference Management in the Semantic Web, 4th Multidisciplinary Workshop on Advances in Preference Handling (M-PREF 2008), AAAI-08, Chicago, USA, July 13-14, 116p.

Thuraisingham, B., 2007, Building Trustworthy Semantic Webs, Auerbach Publications, ISBN:0849350808.

KAYNAKLAR DİZİNİ (devam)

Toninelli, A., Montanari, R., Kagal, L., Lassila, O., 2007, Proteus: A Semantic Context-Aware Adaptive Policy Model, POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, Bologna, Italy, 13–15 June, 129 – 140pp.

Tonti, G., Bradshaw, J. M., Jeffers, R., Monranari, R., Suri, N., Uszok, A., 2003, Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KaoS, Rei, and Ponder, 2nd International Semantic Web Conference (ISWC 2003), 419-437pp.

Uszok, A., Bradshaw, J. M., Jeffers, R., 2004a, KAoS: A Policy and Domain Services Framework for Grid Computing and Semantic Web Services, Second International Conference on Trust Management.

Uszok, A., J. M. Bradshaw, R. Jeffers, A. Tate, J. Dalton, 2004b, Applying KAoS Services to Ensure Policy Compliance for Semantic Web Services Workflow Composition and Enactment, International Semantic Web Conference, 425-440pp.

Yuan, E., Tong, J., 2005a, Attributed Based Access Control (ABAC) for Web Services, In ICWS'05: IEEE International Conference on Web Services, 569p.

KAYNAKLAR DİZİNİ (devam)

Yuan, E., Tong, J., 2005b, Attribute Based Access Control - A New Access Control Approach for Service Oriented Architecture (SOA), New Challenges for Access Control Workshop, Ottawa, ON, Canada, April 27.

EKLER

Ek 1

REI POLİTİKA ONTOLOJİLERİ, SINIFLARI VE ÖZELLİKLERİ

ReiPolicy.owl

Özellikler

- context
- grants
- defaultModality
- defaultBehavior
- rulePriority
- metaDefault
- imports
- createdBy
- creationTime
- modifiedBy
- modifiedTime
- to
- deontic
- requirement

Sınıflar

- ReiRoot
  - Policy
  - Granting
- GrantingOrDeontic
  - Granting

ReiMetaPolicy.owl

Özellikler

- greaterPriority
- lesserPriority

Sınıflar

- MetaPolicy
  - ModalityPrecedence
    - *instance* Positive
    - *instance* Negative
  - Behavior
    - *instance* ExplicitPermImplicitProh
    - *instance* ImplicitPermExplicitProh
    - *instance* ExplicitPermExplicitProh
  - MetaMetaPolicy

        - *instance* CheckModalityFirst
        - *instance* CheckPriorityFirst
    - Priority
        - RulePriority
        - PolicyPriority

ReiEntity.owl

Özellikler

- affiliation
- owner

Sınıflar

- Entity
    - Agent
    - Object
    - Variable

ReiDeontic.owl

Özellikler

- action
- actor

- constraint
    - startingConstraint
    - endingConstraint
- sanction
- obligedTo
- provision

Sınıflar

- DeonticObject
    - Dispensation
    - Obligation
    - Permission
    - Prohibition
- ObligationOrProhibition
    - Obligation
    - Prohibition

ReiAction.owl

Özellikler

- actor
    - sender
- location
- time

- target
- precondition
- effect
- receiver
- condition
- content
- first
- second

Sınıflar

- ActionOrDeontic
    - Action
        - SimpleAction
            - DomainAction
            - Speech Act
                - Delegate
                - Revoke
                - Request
                - Cancel
                - Command
                - Promise
        - CompositeAction
            - Combination
            - Sequence
            - Choice

        - Once
        - Iteration
    - CombOrSeqOrChoice
        - Combination
        - Sequence
        - Choice

ReiConstraint.owl

Özellikler

- subject
- predicate
- object
- first
- second
- equal
- inequal

Sınıflar

- Constraint
    - SimpleConstraint
    - BooleanConstraint
        - And
        - Or

        - Not
    - AndOrOr
        - And
        - Or

ReiAnalysis.owl

Özellikler

- instance
- property
- value
- policy
- granting
- subject
- predicate
- object
- actor
- action,
- target

Sınıflar

- Analysis
    - WhatIf
        - WhatIfProperty

            - WhatIfIAddProperty
            - WhatIfIRemoveProperty
        - WhatIfPolicyRule
            - WhatIfIAddPolicyRule
            - WhatIfIRemovePolicyRule
    - UseCase
        - StatementUseCase
            - TrueStatementUseCase
            - FalseStatementUseCase
        - DeonticUseCase
            - PermissionUseCase
            - ProhibitionUseCase
            - ObligationUseCase
            - DispensationUseCase

Ek 2

KULLANILAN KELİMELER

(Türkçe'den İngilizce'ye)

TÜRKÇE		İNGİLİZCE
Altyapı	:	Infrastructure
Anlatım	:	Expression
Ayrışım	:	Decomposition
Bağlam	:	Context
Betimleme Mantığı	:	Description Logic
Biçim	:	Modality
Bildirim deyimi	:	Declarative
Birim	:	Module
Birimsel	:	Modular
Birleşik giriş kutusu	:	Combo box
Boole işleci	:	Boolean operator
Çelişki	:	Conflict
Çıkarsama	:	Inference
Çıkarsama	:	Reasoning
Değişken	:	Variable
Değişmez	:	Constant
Düğüm	:	Node
Düzenek	:	Mechanism
Erim	:	Range
Eşleme	:	Map
Etki	:	Effect
Etki alanı	:	Domain

Etkileşim	**:**	Interaction
Geçersiz kılmak	**:**	Override
Geçici	**:**	Provisional
Gerçek	**:**	Fact
Gizlilik	**:**	Privacy
Hak	**:**	Right
Havuz	**:**	Repository
Hedef	**:**	Target
İçerik	**:**	Content
İlişkisel	**:**	Relational
İlkel	**:**	Primitive
İptal	**:**	Cancel
İsim uzayı	**:**	Namespace
İsteğe bağlı	**:**	Discretionary
İstek	**:**	Request
İşleç	**:**	Operator
İşlem	**:**	Transaction
İşleyici	**:**	Handler
İzin	**:**	Permission
Karakter dizisi	**:**	String
Kayan	**:**	Float
Kısıt	**:**	Constraint
Kimlik Denetimi	**:**	Authentication
Komut	**:**	Command
Konuşma edimi	**:**	Speech act
Koşul	**:**	Condition
Koşul	**:**	Provision
Kural	**:**	Rule
Nesne Özelliği	**:**	Object Property

Olgu	**:**	Instance
Onaylamak	**:**	Grant
Öncelik	**:**	Precedence
Örgü	**:**	Web
Örnek	**:**	Individual
Örtülü	**:**	Implicit
Özel izin	**:**	Dispensation
Özellik	**:**	Property
Özne	**:**	Subject
Öznitelik	**:**	Attribute
Özyineli	**:**	Recursive
Politika	**:**	Policy
Politika Arıtımı	**:**	Policy Refinement
Politika Deposu	**:**	Policy Store
Sakınım	**:**	Refrain
Seçenek	**:**	Preference
Sıradüzensel	**:**	Hierarchical
Söz verme	**:**	Promise
Süzgeç	**:**	Filter
Tamsayı	**:**	Integer
Tümdengelimli	**:**	Deductive
Uyumlu ek	**:**	Plugin
Uzlaşma	**:**	Negotiation
Üst politika	**:**	Meta policy
Üstünlük, Öncelik	**:**	Priority
Varlık	**:**	Entity
Varsayılan	**:**	Default
Veri türü Özelliği	**:**	Datatype Property
Yapıcı	**:**	Builder

Yaprak	**:**	Leaf
Yaptırım	**:**	Sanction
Yasak	**:**	Prohibition
Yerel	**:**	Native
Yetke	**:**	Authority
Yetki	**:**	Authorization
Yetki	**:**	Clearance
Yetki aktarımı	**:**	Delegate
Yetkinin geri alımı	**:**	Revoke
Yorumlamak	**:**	Interpret
Yöneltici	**:**	Router
Yürütme	**:**	Enforcement
Yürütme	**:**	Execute
Zorunlu	**:**	Mandatory
Zorunluluk	**:**	Obligation

Ek 3

KULLANILAN KELİMELER

(İngilizce'den Türkçe'ye)

İNGİLİZCE		TÜRKÇE
Attribute	**:**	Öznitelik
Authentication	**:**	Kimlik Denetimi
Authority	**:**	Yetke
Authorization	**:**	Yetki
Boolean operator	**:**	Boole işleci
Builder	**:**	Yapıcı
Cancel	**:**	İptal
Clearance	**:**	Yetki
Combo box	**:**	Birleşik giriş kutusu
Command	**:**	Komut
Condition	**:**	Koşul
Conflict	**:**	Çelişki
Constant	**:**	Değişmez
Constraint	**:**	Kısıt
Content	**:**	İçerik
Context	**:**	Bağlam
Datatype Property	**:**	Veri türü Özelliği
Declarative	**:**	Bildirim deyimi
Decomposition	**:**	Ayrışım
Deductive	**:**	Tümdengelimli
Default	**:**	Varsayılan
Delegate	**:**	Yetki aktarımı

Description Logic : Betimleme Mantığı
Discretionary : İsteğe bağlı
Dispensation : Özel izin
Domain : Etki alanı
Effect : Etki
Enforcement : Yürütme
Entity : Varlık
Execute : Yürütme
Expression : Anlatım
Fact : Gerçek
Filter : Süzgeç
Float : Kayan
Grant : Onaylamak
Handler : İşleyici
Hierarchical : Sıradüzensel
Implicit : Örtülü
Individual : Örnek
Inference : Çıkarsama
Infrastructure : Altyapı
Instance : Olgu
Integer : Tamsayı
Interaction : Etkileşim
Interpret : Yorumlamak
Leaf : Yaprak
Mandatory : Zorunlu
Map : Eşleme
Mechanism : Düzenek
Meta policy : Üst politika
Modality : Biçim

Modular	**:**	Birimsel
Module	**:**	Birim
Namespace	**:**	İsim uzayı
Native	**:**	Yerel
Negotiation	**:**	Uzlaşma
Node	**:**	Düğüm
Object Property	**:**	Nesne Özelliği
Obligation	**:**	Zorunluluk
Operator	**:**	İşleç
Override	**:**	Geçersiz kılmak
Permission	**:**	İzin
Plugin	**:**	Uyumlu ek
Policy	**:**	Politika
Policy Refinement	**:**	Politika Arıtımı
Policy Store	**:**	Politika Deposu
Preference	**:**	Seçenek
Precedence	**:**	Öncelik
Priority	**:**	Üstünlük, Öncelik
Primitive	**:**	İlkel
Privacy	**:**	Gizlilik
Prohibition	**:**	Yasak
Promise	**:**	Söz verme
Property	**:**	Özellik
Provision	**:**	Koşul
Provisional	**:**	Geçici
Range	**:**	Erim
Reasoning	**:**	Çıkarsama
Recursive	**:**	Özyineli
Refrain	**:**	Sakınım

Relational	**:**	İlişkisel
Repository	**:**	Havuz
Request	**:**	İstek
Revoke	**:**	Yetkinin geri alımı
Right	**:**	Hak
Router	**:**	Yöneltici
Rule	**:**	Kural
Sanction	**:**	Yaptırım
Speech act	**:**	Konuşma edimi
String	**:**	Karakter dizisi
Subject	**:**	Özne
Target	**:**	Hedef
Transaction	**:**	İşlem
Variable	**:**	Değişken
Web	**:**	Örgü

ÖZGEÇMİŞ

Soyadı	: Can
Adı	: Özgü
Doğum Tarihi	: 10.07.1978
Doğum Yeri	: Manisa
Uyruk	: T.C.
Adres	: Ege Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova/İZMİR
Email	: ozgu.can@ege.edu.tr
Ev Telefonu	: 0 – 232 – 339 30 29
İş Telefonu	: 0 – 232 – 343 40 00 / 5309
Cep Telefonu	: 0 – 532 – 708 33 31
Yabancı Dil	: İngilizce, Almanca, Fransızca
Eğitim	:
1999 – 2003	Yüksek Lisans, Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, Ağ Teknolojileri Bilim Dalı.
2002 – 2006	Lisans, Anadolu Üniversitesi, İktisadi ve İdari Bilimler Fakültesi, İşletme Bölümü.
1995 – 1999	Lisans, Selçuk Üniversitesi, Bilgisayar Mühendisliği Bölümü.
1988 – 1995	Lise, Manisa Fatih Anadolu Lisesi.