Yeniden Kullanılabilir Yazılım Bileşenlerinin Tanımlanması için Ontoloji Tabanlı Bileşik Bir Model

Rıza Cenk Erdur[1], Oylum Alatlı[1] ve Özgün Yılmaz[1]

[1] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye.

{cenk.erdur,oylum.alatli,ozgun.yilmaz}@ege.edu.tr

Özet. RIG-BIDM ve Reboot, yeniden kullanılabilir yazılım bileşenlerinin tanımlanmasında sıkça kullanılan iki modeldir. Her iki model de yazılım bileşenlerini farklı yönlerden tanımlamaktadır. Bu iki modelin birleştirilmesi ile oluşturulacak bileşik bir model bileşenlerin daha net bir biçimde tanımlanmasını olanaklı kılacaktır. Bu çalışmada RIG-BIDM ve Reboot modelleri bu doğrultuda birleştirilerek bileşik bir bileşen modeli oluşturulmuştur. Anlamsal Web'in makine çıkarsaması ve SPARQL sorguları gibi teknolojik olanaklarından faydalanabilmek amacıyla oluşturulan model OWL dilinde geliştirilmiş bir ontoloji olarak ifade edilmiştir. Bu ontolojinin kullanımına bir örnek vermek amacıyla, Anlamsal Web üzerinde çalışan açık kaynaklı bir sorgu motorunun içerdiği yazılım bileşenlerinin bazılarının tanımlamaları yapılmıştır.

Anahtar Kelimeler: Yeniden Kullanılabilir Yazılım Bileşeni, Bileşen Modeli, Ontoloji.

An Ontology Based Combined Model for Defining Reusable Software Components

Abstract. RIG-BIDM and Reboot are two of the widely used models for defining reusable software components. Both of these models define software components from different perspectives. A combined model, which will be formed by combining these two models, will make it possible to define components in a more comprehensive way. In this respect, in this study, a combined component model has been defined using RIG-BIDM and Reboot models. In order to make use of the technical opportunites such as machine inference and SPARQL queries provided by the Semantic Web, the defined model has been represented as an OWL ontology. In order to give an example for the usage of this ontology, some components of an open source query engine that runs on the Sematic Web are defined.

Keywords: Reusable Software Component, Component Model, Ontology

1 Giriş

Yeniden kullanım-tabanlı yazılım geliştirme, yeni yazılımların geliştirilmesi sürecinde var olan yazılım bileşenlerini kullanmaya dayalı olup, yazılım üretim ve bakım maliyetini düşüren, hızlı geliştirmeyi destekleyen ve kaliteyi artıran bir yaklaşım olarak kabul görmüştür [1, 2]. Yazılım bileşenleri yalnızca kaynak kod içeren birimler değil, aynı zamanda yazılım ile ilgili kavramsal bilgileri içeren birimler de olabilmektedir. Bu doğrultuda, kaynak kodların yanı sıra analiz ve tasarım modelleri, tasarım desenleri, test senaryoları gibi yazılım yaşam döngüsünün her aşamasından gelen ürünler yeniden kullanılabilir yazılım bileşeni olarak tanımlanabilmektedir.

Bir kuruluşta sistematik yeniden kullanımın gerçekleşmesi için, kuruluşta var olan yazılım geliştirme sürecinin yeniden kullanıma dayalı geliştirmeyi destekleyecek şekilde uyarlanması gerekmektedir. Kullanıcıların, gerek kuruluş içindeki gerekse dışsal bileşen kütüphanelerinden gereksinimlerini en iyi karşılayacak yeniden kullanılabilir yazılım bileşenlerini arayıp bulmaları bu tür bir sürecin önemli aşamalarından birisidir. Bu keşif süreci sonunda kullanıcılar buldukları bileşenleri ya olduğu gibi kullanarak ya da uyarlayarak geliştirmekte oldukları sistem ile tümleştirmektedirler.

Bileşen modeli, bir yazılım bileşeninin temel özelliklerini, içsel yapısını, bağımlılıklarını belirleyen bir modeldir. Diğer bir deyiş ile bir bileşen modeli bileşenler için üst-veri tanımlamayı olanaklı kılmaktadır. Bu yönü ile bileşen modeli bileşen arama ve bulma aşamasında kritik bir rol oynamaktadır. Literatürde bileşenleri farklı perspektiflerden tanımlamak için kullanılabilecek değişik bileşen modelleri önerilmiştir. Bu çalışmada ilk olarak literatürde en çok kullanılan modellerden iki tanesi birleştirilerek, bileşenleri daha iyi tanımlayabilecek bir bileşik bileşen modeli oluşturulmuş, daha sonra Anlamsal Web'in makine çıkarsaması ve SPARQL sorguları gibi teknolojik olanaklarından faydalanabilmek amacıyla oluşturulan bileşik model OWL dilinde geliştirilmiş bir ontoloji olarak ifade edilmiştir. Önerilen ontolojinin kullanımının örneklenmesi amacı ile bölümümüzde daha önce geliştirilen, Anlamsal Web üzerinde çalışan açık kaynaklı bir bağlı veri sorgu motorunun içerdiği bazı yazılım bileşenleri, bu ontoloji ile tanımlanmış ve sorgulama örnekleri verilmiştir. Örnekte kullanılan bağlı veri sorgu motorunun Anlamsal Web ortamında çalışmasının, bu bildiride önerilen ontoloji ile bir ilgisinin olmadığını ve önerilen ontoloji ile Anlamsal Web üzerinde çalışıp çalışmadığına bakılmaksızın herhangi bir yazılım sistemine ilişkin bileşenlerin tanımlanabileceğini vurgulamak isteriz.

Bildirinin izleyen kısımları şu şekilde düzenlenmiştir: Bölüm 2'de bu çalışmada kullanılan bileşen modelleri tanıtılmaktadır. Bölüm 3'de ilgili çalışmalar verilmiştir. Bölüm 4, geliştirilen bileşik model ve OWL ontolojisini tartışmaktadır. Bölüm 5, geliştirilen ontolojinin kullanımını örneklemektedir. Sonuçlar, Bölüm 6'da verilmektedir. Son olarak kaynaklar listelenmektedir.

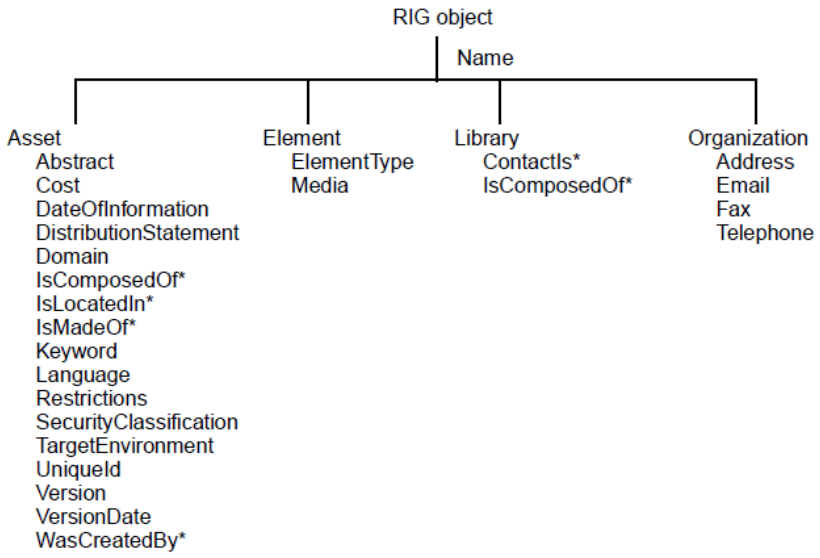
2 Bileşen Modelleri

Bileşen modelleri yeniden kullanılabilir bileşenler için üst-bilgi tanımlanmasını olanaklı kılmaktadır. Tanımlanan üst-bilgiler de, yeniden kullanılabilir bileşenlerin kul-

lanıcılar tarafından daha iyi anlaşılmasını sağlayacak önemli bir unsurdur. Tipik bir bileşen üst-bilgisi içinde; bileşenin adı, hangi uygulama alanı için hangi dil kullanılarak geliştirildiği, sürümü, nasıl yeniden kullanılabileceği/uyarlanabileceğine ilişkin bilgiler, belli bir sınıflandırma yöntemine göre sınıflandırılmış ise buna ilişkin anahtar kelimeler, hangi kişi ve/veya organizasyon tarafından geliştirildiği, ücreti, sürümü gibi bilgiler yer almaktadır.

Bileşen üst-bilgisi tanımlamak için farklı projeler kapsamında çeşitli şablonlar önerilmiştir. REBOOT (REuse Based on Object Oriented Techniques) [3, 4] projesi kapsamında önerilen şablon, özelliklere (facet) dayalı sınıflandırma yaklaşımına [5] dayanan sınıflandırma bilgisini oluşturan anahtar kelimeler yanı sıra bileşenin oluşturulduğu tarih, oluşturan kişi kuruluş, sürüm, bileşenin geliştirildiği araç ve proje, varsa geçmiş deneyimler gibi bilgileri de içermektedir. Sınıflandırma amacı ile dört adet özellik (facet) tanımlanmıştır. Dördüncü bölümde daha ayrıntılı söz edileceği üzere, bu özellikler bileşene ilişkin soyutlama, bileşenin desteklediği işlemler, üzerinde işlem yaptığı veri tipi veya nesneler ve bağımlılıklar olarak adlandırılmaktadır. Her bir özellik kendi içinde bir terim uzayına sahip olup, bu uzaylardan seçilecek terimlerin bir araya gelmesi ile bileşene ilişkin sınıflandırma bilgisi oluşmaktadır.

Bileşen üst-bilgisi tanımlamaya yönelik diğer bir şablon ise RIG (Reuse library Interoperability Group) tarafından önerilen ve BIDM (Basic Interoperability Data Model- Temel Birlikte-İşlerlik Veri Modeli) olarak adlandırılan şablondur. Bu model, IEEE standardı olarak kabul edilmiştir [6]. RIG, veri modeli bileşenler için İngilizce "asset" kelimesini kullanmaktadır. BIDM, bir sınıf sıradüzeni olarak tanımlanmıştır ve RIG nesnesi en genel öğe olarak en üstte yer almaktadır. RIG nesnesinin altında dört adet sınıf bulunmaktadır. Bu sınıflar; bileşen(asset), eleman(element), kütüphane(library) ve organizasyon(organization) olarak adlandırılmıştır [6]. Her bir sınıf altında farklı öznitelikler yer almaktadır. Bu özniteliklerden bazıları sınıflar arasındaki ilişkileri tanımlamaktadır. Örneğin, "asset" altındaki "wasCreatedBy" özniteliği "asset" ile "organization" arasında bir ilişki kurmaktadır. RIG-BIDM modeli Şekil 1'de görülmektedir.



Şekil 1. RIG-BIDM bileşen modeli ([6] numaralı kaynaktan alınmıştır.)

3 İlgili Çalışmalar

Anlamsal Web teknolojilerinin yazılım yeniden kullanımı alanında kullanılmasını tartışan çeşitli çalışmalar literatürde yer almaktadır. Bu bölümde bu çalışmalar kısaca değerlendirilecektir.

KOntoR olarak adlandırılan altyapı [7], bileşen üst-bilgilerine dayanan yeniden kullanım yaklaşımını önermekte ve üst-bilgilerin tanımlanması için bir şablon önermektedir. Bu çalışmada önerilen şablon, yeniden kullanılabilir bileşenlerin tanımlanabilmesi için belli yapıları sunmaktadır ancak RIG-BIDM veya diğer standartlara dayalı değildir, sınıflandırma bilgisi içermemektedir ve önerilen şablonun ontoloji olarak ifade edilmiş karşılığı veya buna bir bağlantı bulunmamaktadır.

Alnusair ve Zhao, bileşen yeniden kullanımı için bir ontoloji modeli önermektedir [8]. Bu ontoloji kaynak kod odaklı olup, nesneye dayalı yaklaşım ile geliştirilmiş kaynak kodların metot, sınıf, arayüz seviyesinde tanımlanmasına yöneliktir ve standartları göz önüne almamaktadır.

Nianfang ve arkadaşları, ilk olarak ADA dili bileşenlerini tanımlamak için ortaya konan 3C modelini temel alan bir ontoloji tanımlamıştır [9]. Ancak bildiride verilen ontoloji tüm modeli göstermemekte ve ontolojinin tamamının görülebileceği bir bağlantı verilmemektedir.

Bileşenlerin temsili ve geri elde edilmesinde Anlamsal Web teknolojileri kullanan diğer bir çalışmada önerilen bileşen ontolojisi model olarak verilmiştir [10]. Bu model oldukça kapsamlı olmasına rağmen bir bileşenin diğer bileşenlerden oluşma durumu-

nu dikkate almamaktadır. Ayrıca, modelin ontoloji dili ile ifadesi veya buna bir bağlantı bulunmamaktadır.

Günümüzde çok popüler olan Maven aracında kullanan Project Object Model (POM) de yazılım yeniden kullanılabilirliği açısından önem taşımaktadır. Bir derleme otomasyonu ve proje yönetim aracı olan Apache Maven, POM kavramı üzerine oturtulmuştur. POM bir XML (Extensible Markup Language) dosyasında tanımlanır. Proje ve konfigürasyon detayları ile ilgili bilgileri içerir. Proje bağımlılıkları, gerekli eklentiler, versiyon, geliştiriciler, vs. POM ile tanımlanabilmektedir [14, 15]. Maven POM bir projenin koddan oluşan bileşenleri ile bu bileşenlerin arasındaki bağımlılıkların betimlenmesine odaklanmaktadır ve ontoloji tabanlı değildir. Bu bildiride incelenen çalışmada ise günümüzde yazılım geliştirme yaşam döngüsünün parçası kabul edilen tüm bileşenler (UML diyagramları, diğer diyagramlar, kullanım senaryoları (use case), yazılım sınama bileşenleri, dokümantasyon kısımları, vs.) yeniden kullanım için modellenebilmektedir. Dahası, Maven'in yüzeysel bileşen tanımlamalarının aksine gerçekleştirilen çalışma ile yazılım bileşenlerinin arasındaki bağımlılıklar yazılımda kullanılan operantlar düzeyine kadar detaylandırılabilmektedir. Bu çalışmanın Maven'den ayrılan bir diğer özelliği ise dil bağımsız bir model hedeflemesidir. Oysa, Maven POM dosyalarında Java dilindeki yeniden kullanılabilir bileşenlerin modellenmesi hedeflenmektedir.

Anlamsal Web teknolojilerini yazılım yeniden kullanımı alanında kullanan yakın tarihli bir çalışmada da kaynak kod içeren bileşenler anlamsal olarak ifade edilmiş olup, bu anlamsal bilgi kullanıcılara gereksinimlerine göre bileşen önermede kullanılmaktadır [11].

Anlamsal bilgiye dayanarak bileşen önerme gibi modülleri geliştirmek ilk aşamada bu bildirinin odağında yoktur; var olan standartlar özümsenerek yeniden kullanılabilir bileşenler için ontoloji tabanlı bileşik bir model önermek ana hedeftir. Diğer taraftan önerilen model bileşen önerme de dahil olmak üzere farklı sistemlerde kullanılabilecektir.

4 Geliştirilen Model ve Ontoloji

Bu bildiri kapsamında geliştirilen yazılım bileşeni ontolojisinde daha önce tanıtılmış olan RIG-BIDM standardı ile sıklıkla kullanılan REBOOT şablonu birleştirilerek her iki modeli de içerecek geniş kapsamlı bir model oluşturulmaya çalışılmıştır. Böylelikle her iki modelin de birbirini tamamlayıcı rol oynaması hedeflenmiştir.

Örneğin, REBOOT modelinde bileşenlerin içerdikleri operasyonlar ve bu operasyonların operantları tanımlanmakta, ancak bu tanımlamalar RIG-BIDM modelinde yapılamamaktadır. Benzer şekilde RIG-BIDM modelinde bir yazılım bileşenini meydana getiren elemanlar ve bileşenler, bileşenlerin ait oldukları kütüphaneler ve bu kütüphaneleri meydana getiren diğer kütüphane ve bileşenler tanımlanabilirken, bu tanımlamalar REBOOT modelinde yapılamamaktadır.

Bileşik modelin oluşturulabilmesi amacıyla öncelikle REBOOT modeli bir OWL ontolojisi ile ifade edilmiş; ardından RIG-BIDM modelinin ifade edildiği bir ontolojinin içinde kullanılarak her iki model birleştirilmiştir. Bu bölümde öncelikle

REBOOT ve RIG-BIDM modellerinin ifade edildiği ontolojiler incelenecek, ardından geliştirilen bileşik model tanıtılacaktır.

4.1 REBOOT Ontolojisi

REBOOT (REuse Based on Object Oriented Techniques) yaklaşımının amacı, yazılımın yeniden kullanımını teşvik ederek ve destekleyerek, yazılım geliştirmede verimliliği ve kaliteyi arttırmaktır. REBOOT iki yazılım geliştirme süreci modelini (yeniden kullanım için geliştirme ve yeniden kullanım ile geliştirme) içerir. Bu modellerde yeniden kullanıma özgü etkinlikler geliştirme yaşam döngüsünün tüm aşamalarına katılır. Ayrıca bir dizi araç ve REBOOT ortamı, bu iki modelin desteklenmesine ve kuruluşların yeniden kullanılabilir bileşenleri yönetmesine yardım amacıyla geliştirilmiştir [3].

REBOOT dört önemli özelliğe göre sınıflandırma yapar [4]:

- **Soyutlama:** Nesneye yönelik programlamada bir bileşen genellikle bir isim veya isim tamlamasıyla (Ör: takvim, uçuş yöneticisi, yangın alarm sistemi, vs.) nitelendirilebilir.

- **İşlemler:** Bileşenlerin yaptığı işlemleri, diğer bir deyişle sağladığı fonksiyonları içeren özelliktir. Örneğin bir yığıt bileşeni tepeye öğe ekleme ve tepeden öğe çıkarma işlemlerini sağlamalıdır.

- **İşlem yapılanlar:** Bu özellik, bileşenin hangi veri tipi veya nesne üzerinde işlem yaptığını belirtir. (Ör: tam sayı, liste, kaynak)

- **Bağımlılıklar:** Bileşenin çalışabilmesi için gerekli ortamı veya diğer bileşenleri belirtir. Bir bileşen başka bileşenlere, kütüphanelere, programlama diline veya işletim sistemine bağımlı olabilir.

Geliştirilen ontolojide, yeniden kullanılabilir bileşen ve bu dört özellik OWL sınıfları ile modellenmiştir. Bileşen, bu dört özellik ile nesne özellikleri (object property) aracılığıyla ilişkilendirilmiştir. Soyutlama sınıfının altında isimler bir hiyerarşi oluşturacak şekilde gruplanmıştır. Örneğin, yığıt terimi sırasıyla, soyutlama, veri yapıları, koleksiyon, sıralı olmayan koleksiyon sınıflarının alt sınıfıdır. Bağımlılıklar da hiyerarşi oluşturacak şekilde yapılandırılmıştır. Bağımlılık sınıfının alt sınıflarını kütüphane, çerçeve, programlama dili ve işletim sistemi sınıfları oluşturmaktadır. Bu alt sınıfların altında tanımlanan sınıflar bir bağımlılık hiyerarşisi oluşturmaktadır. İşlem yapılanlar sınıfının altında ise ilkel veri tipleri (tamsayı, ondalıklı sayı, boolean, vs.), karmaşık veri tipleri (sınıflar) ve diğer kaynaklar tanımlanmıştır. İşlem sınıfı ise bileşen tarafından sağlanan işlevler ile ilişkilendirilmiştir.

4.2 RIG-BIDM Ontolojisi

RIG-BIDM yeniden kullanılabilir kütüphanelerin içerdikleri bileşenlerin tanımlanmasında kullanılacak minimal bilgi kümesini tanımlamayı amaçlayan IEEE tarafından standartlaştırılmış bir modeldir [6].

BIDM modelinde yeniden kullanılabilir bileşenler varlık ya da kıymetli şey anlamına gelen "asset" kelimesi ile ifade edilmektedir. Her varlık da başka varlıklardan ya da "element" kelimesi ile ifade edilen daha küçük birimler olan elemanlardan oluşabilir. Örneğin, bir Java paketi ve bu paketin içerdiği diğer paketler birer varlık, paketin içerdiği sınıflar ise elemanlar olarak nitelenebilir. Varlıkların bir araya gelmesiyle ise kütüphaneler oluşmaktadır. Bir kütüphane başka kütüphaneler de içerebilir.

Bu tanımlamalar BIDM modelinde bir sıradüzen içerisinde verilmektedir. Sıradüzenin en üstünde RIG nesnesi yer almaktadır. RIG nesnesi diğer dört temel sınıf olan varlık ("asset"), eleman ("element"), kütüphane ("library") ve organizasyon ("organization") sınıflarının türetildiği sınıftır. Organizasyon sınıfı varlıkları oluşturan organizasyonların tanımlanması amacıyla kullanılmaktadır.

Geliştirilen BIDM ontolojisinde RIG nesnesi ile diğer dört temel sınıf BIDM modelinin içerdiği orijinal terimlerle isimlendirilmiş birer OWL sınıfı ile ifade edilmiştir. Her OWL sınıfı BIDM modelinde verilen sınıf özelliklerini içerecek şekilde tanımlanmıştır. Bu özellikler 2. bölümdeki Şekil 1'de görülebilir. "Asset" sınıfı modelden farklı olarak bir programlama dilinde ifade edilmiş yazılımları ve doğal dil ile ifade edilmiş dokümanları ayıracak şekilde "PLAsset" ve "NLAsset" olarak iki altsınıfa ayrılmıştır.

4.3 REBOOT ve RIG-BIDM Ontolojilerinin Birleştirilmesi

REBOOT ve RIG-BIDM modellerinin birleştirilmesi amacıyla SCO ("Software Component Ontology") isimli ontoloji tanımlanmıştır. SCO RIG-BIDM ontolojisinin REBOOT ontolojisini içerecek şekilde genişletilmesi ile oluşturulmuştur.

REBOOT ve RIG-BIDM ontolojilerinin ortak noktaları yazılım bileşenlerinin tanımlandığı "asset" (varlık) ve "component" (bileşen) sınıflarıdır. Bu nedenle SCO ontolojisinde bu iki sınıf denk sınıflar olarak tanımlanarak bir bileşen için iki ontolojide ayrı ayrı verilen bilgilerin birleştirilebilmesi sağlanmıştır.

Bileşenlerin işlevlerinin anlatıldığı soyutlama ifadeleri de her iki modelde ortak olmasına karşın, BIDM modelinde varlık sınıfının bir özelliği iken REBOOT modelinde bileşenin sahip olduğu soyutlama tipini ifade eden bir sınıf olarak tanımlanmıştır. Soyutlama sınıfı REBOOT ontolojisinde "Abstraction" olarak adlandırılmıştır. Her iki modeldeki soyutlama ifadelerinin aynı bileşene ait soyutlamanın ifade edilmesinde kullanılabilmesi için soyutlama BIDM ontolojisinde de "Abstract" şeklinde isimlendirilen bir sınıf olarak tanımlanmış ve karakter dizisi tipinde tanımlanmış "abstractionDescription" (soyutlama tanımı) veri özelliği ("data property") ile "asset" (varlık) sınıfı ile ilişkilendirilmiştir. Bir bileşeni tanımlayan "Abstract" ve "Abstraction" sınıfı örneklerinin "sameAs" özelliği ile bağlanması ile her iki modelden gelen soyutlama ifadelerinin birleştirilmesi mümkün olacaktır.

4.4 Durum Çalışması Örneği: WoDQA Federe Sorgu Motorunun SCO ile Tanımlanması

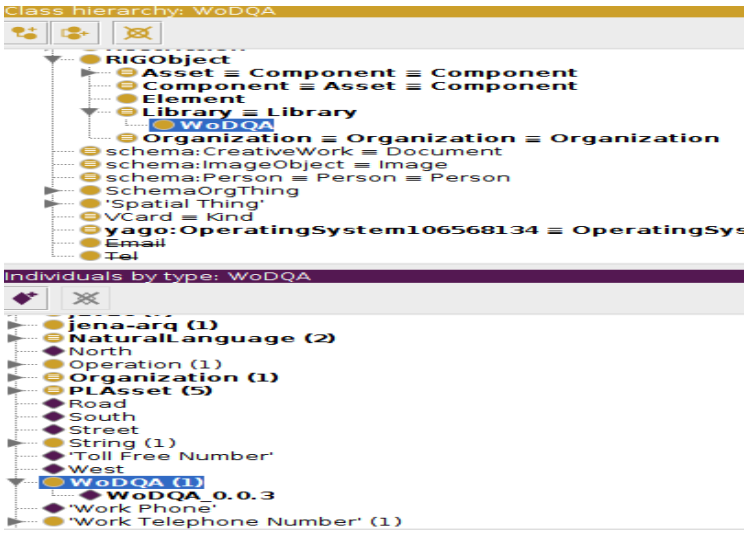
Anlamsal Web çok sayıda birbiri ile bağlantılı veri setinden oluşmaktadır. Bu veri setleri üzerinde çalıştırılan bazı sorgular birden fazla veri setinden getirilecek verilerin

birleştirilmesini gerektirmektedir. Bu tip sorgulara federe sorgular adı verilmektedir. Federe sorguların çalıştırılabilmesi için ise federe sorgu motorlarına ihtiyaç duyulmaktadır. Bu durum çalışmasında incelenecek olan WoDQA (Web of Data Query Analyzer) [12] da Anlamsal Web üzerinde çalıştırılmak istenen federe SPARQL sorgularını işletebilen bir federe sorgu motoru olarak tanımlanabilir.

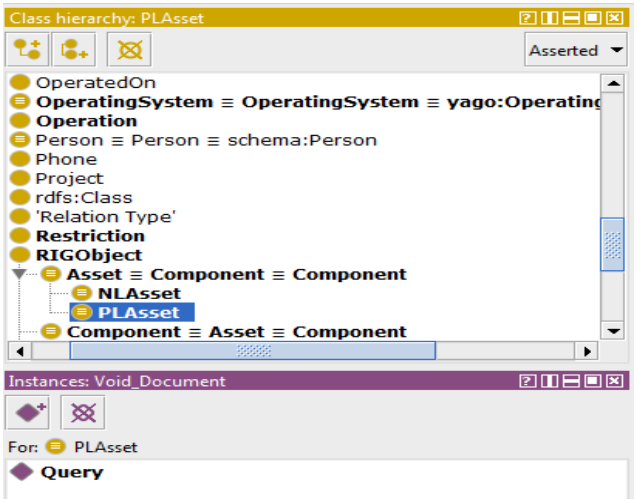
WoDQA bir federe sorgu motoru olmasının yanı sıra yeniden kullanılabilir yazılım bileşenleri içeren bir kütüphane olarak da kullanılabilmektedir. Dolayısıyla, Şekil 2'de de görüldüğü gibi WoDQA SCO ontolojisinde "Library" sınıfının bir altsınıfı, WoDQA'nın 0.0.3 numaralı versiyonu ise WoDQA altsınıfının bir örneği olarak tanımlanmıştır.

WoDQA kütüphanesi birçok bileşen içermektedir. Burada daha net ve kolay anlaşılır bir örnek olması amacıyla, Anlamsal Web üzerindeki sorguları ifade etmek ve işletmek için kullanılan Query bileşeni incelenecektir. Query bileşeni denk sınıflar olan "Asset" ve "Component" sınıfınlarından bir örnek olan tanımlanmıştır. İlgili tanımlamalar Şekil 3'te görülebilir.

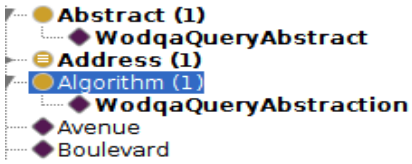
Query bileşeninin işlevinin tanımlanması amacıyla, BIDM ontolojisindeki "Abstract" sınıfının örneği olan "WodqaQueryAbstract" ve REBOOT ontolojisindeki "Abstraction" sınıfının altsınıfı olan "Algorithm" (algoritma) sınıfının örneği olan "WodqaQueryAbstraction" ifadeleri tanımlanmıştır. İlgili tanımlamalar Şekil 4'te görülebilir. Her iki ifadenin de aynı soyutlamaya ait olduğu Şekil 5'te görülen "sameAs" bağlantısı ile ifade edilmektedir. Yine Şekil 5'te WodqaQueryAbstract örneğinin "abstractDescription" veri özelliği ile tanımlanması görülebilir.



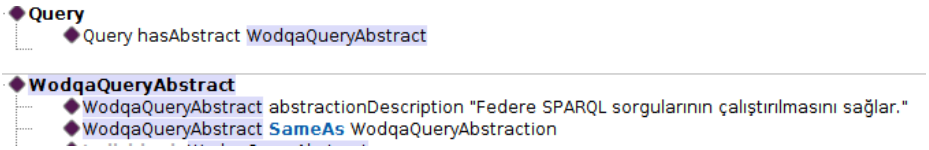
Şekil 2. WoDQA kütüphanesinin SCO ontolojisindeki tanımlaması



Şekil 3. WoDQA kütüphanesindeki Query bileşeninin SCO ontolojisindeki tanımlaması



Şekil 4. WoDQA kütüphanesindeki Query bileşeninin soyutlamasının SCO ontolojisindeki tanımlaması



Şekil 5. WoDQA kütüphanesindeki Query bileşeninin soyutlamalarının "sameAs" bağlantısı ile birleştirilmesi ve WodqaQueryAbstract örneğinin soyutlama tanımlaması

Şekil 6'da Query bileşenine ait tüm nesne özellikleri görülebilir. Bu şekilde görülen özellikler BIDM ve REBOOT modellerinin birbirlerini tamamlayıcı özelliklerini net bir şekilde göstermektedir.

Şekil 6'da görülebilen tanımlamalardan BIDM ontolojisinden gelenler şunlardır:

- Query bileşeninin içinde bulunduğu kütüphaneyi tanımlayan "locatedIn" özelliği,
- bileşenin ait olduğu alanı tanımlayan "hasDomain" özelliği,
- bileşeni oluşturan öğeleri ifade eden "madeOf" özelliği ile yapılmış tanımlamalar,

- bileşenin gerçekleştirildiği dili tanımlayan “inProgrammingLanguage” özelliği,
- bileşenin üzerinde çalışmak için ihtiyaç duyduğu ortamı BIDM ontolojisindeki “environment” sınıfının bir örneği olan “WodqaQueryEnvironment” ile tanımlayan “hasTargetEnvironment” özelliği,
- “WodqaQueryEnvironment” örneğinin “needsCompiler” özelliğinin yine BIDM ontolojisindeki “Compiler” sınıfının bir örneği olan javac_1.7 ile tanımlanması.

Şekil 6'da görülebilen tanımlamalardan REBOOT ontolojisinden gelenler ise şunlardır:

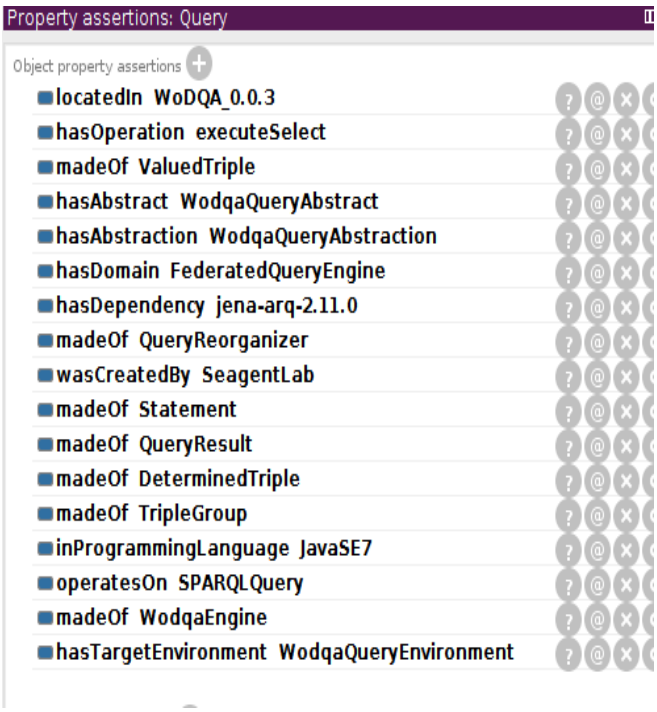
- bileşenin sahip olduğu “executeSelect” operasyonunu tanımlayan “hasOperation” özelliği,
- bileşenin gereksinim duyduğu Jena-arq kütüphanesini tanımlayan “hasDependency” özelliği,
- bileşenin sahip olduğu “executeSelect” operasyonunun operantlarını tanımlayan “operatesOn” özelliği,

Yukarıdaki tanımlamalardan da görüldüğü üzere, BIDM ontolojisi REBOOT ontolojisini; REBOOT ontolojisi de BIDM ontolojisini tamamlayıcı rol oynamaktadır. BIDM ontolojisinde yapılabilen locatedIn, hasDomain, vb. gibi tanımlamalar REBOOT ontolojsinde yapılamamaktadır. Benzer şekilde REBOOT ontolojisinde yapılabilen hasOperation, hasDependency, vb. gibi tanımlamalar da BIDM ontolojsinde yapılamamaktadır. İki ontolojinin birleştirilmesi yazılım yeniden kullanılabilirliği açısından daha yüksek ifade gücüne sahip bir ontoloji oluşturulmasını olanaklı kılmıştır.

Geliştirilen ontolojinin bir diğer olumlu özelliği ise, sağladığı ifade gücünün Anlamsal Web üzerinde kullanılan SPARQL sorgulama dili ile daha da etkili şekilde kullanılabilmesidir. Örneğin, WoDQA kütüphanesinden habersiz bir yazılım geliştirici, FederatedQueryEngine (sco:FederatedQueryEngine) alanına ait kütüphaneler arasından SPARQL sorguları (sco:SPARQLQuery) üzerinde çalışan (ro:operatesOn) bir bileşen aradığında, Şekil 7'de görülen SPARQL sorgusu ile bileşik SCO ontolojisini sorgulayabilir. Bu sorgunun sonucu ise yine Şekil 7'de görülmektedir.

Geliştirilen SCO ontolojisi daha bir çok veri ve nesne özellikleri ile sınıflar içermektedir. Bu bildiri kapsamında en önemli görülen özellik ve sınıflar tanıtılmıştır. SCO ontolojisinin ve bu bildiride incelenen durum çalışmasının detayları ontoloji dipnot[1] olarak verilen bağlantıdan indirilerek incelenebilir.

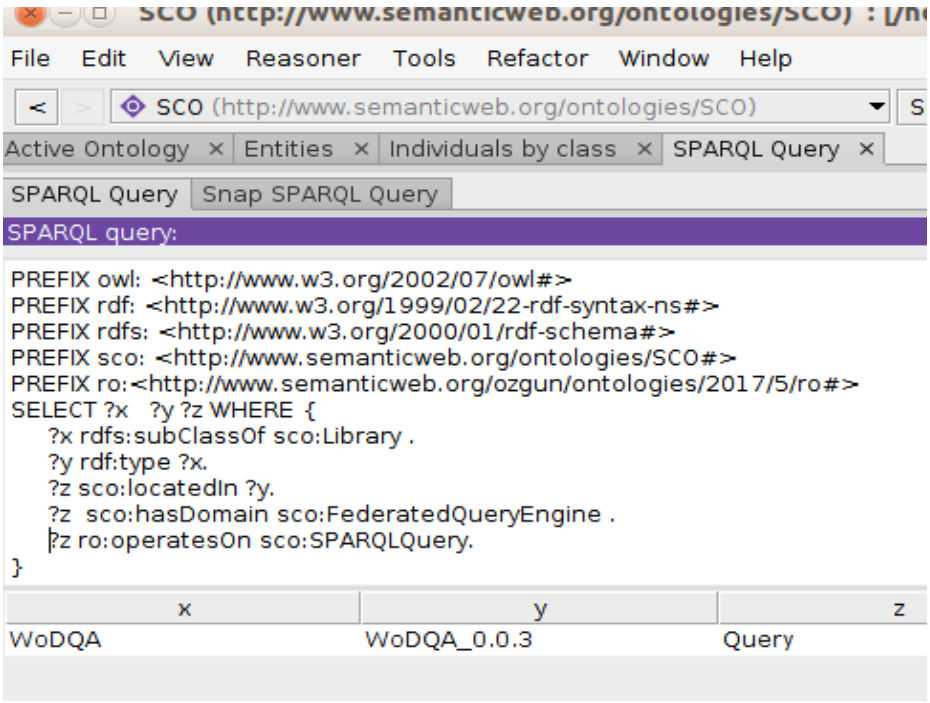
[1] https://webprotege.stanford.edu/#projects/525baa5b-a201-4ca7-9ee7-bdd6de8c2e20/edit/Classes



Şekil 6. WoDQA kütüphanesindeki Query bileşeninin nesne özellikleri

5 Sonuçlar ve Planlanan Çalışmalar

Bu çalışmada standartlaşmış BIDM bileşen modeli ile sıklıkla kullanılan REBOOT bileşen modeli birleştirilerek daha zengin bir bileşen modeli oluşturulmaya çalışılmıştır. Geliştirilen model ile tanımlanacak bileşen kütüphanelerinin yazılım yeniden kullanımında önemli bir adım olan bileşen tanımlama ve arama süreçlerini daha verimli hale getireceği öngörülmektedir. Zira durum çalışmasında da görüldüğü üzere bileşenlerin ve onları oluşturan öğelerin sadece BIDM ya da sadece REBOOT modelleri ile ifade edilmesi yeterli olmamakta, her model bileşen tanımlamasında diğer modelde bulunmayan yönlerden katkı sağlamaktadır. Örneğin BIDM modeli bir bileşeni meydana getiren öğeleri tanımlayabilirken, REBOOT modeli bu tanımlamadan yoksundur. Benzer şekilde REBOOT modeli bir bileşenin sahip olduğu operasyonları ve bu operasyonların kullandıkları operantları tanımlayabilirken BIDM modeli bu tanımlamadan yoksundur. Özetle her bir model diğerini değişik yönlerden tamamlamaktadır. Bu çalışmanın ilerleyen safhalarında 3C modeli [13] gibi geniş kabul gören diğer modeller de ontolojik olarak ifade edilerek geliştirilen bileşik modelin ve SCO ontolojisinin zenginleştirilmesi planlanmaktadır.



Şekil 7. WoDQA kütüphanesine REBOOT ve BIDM ontolojilerinin her ikisinden de gelen özellikleri kullanarak ulaşan sorgu ve sonucu

Referanslar

1. Frakes, W.B. and Kang, K..: Software reuse research: status and future. IEEE Transactions on Software Engineering 31(7), 529-536 (2005).
2. Sommerville, I.: Software engineering. 10th ed., Chapter 15. Pearson, U.S.A. (2016).
3. Morel, J.M., Faget, J.: The REBOOT environment. In: Advances in Software Reuse, pp. 80-88. (1993).
4. Sindre, G., Conradi, R., Karlsson, E.-A.: The REBOOT approach to software reuse. Journal of Systems and Software 30, 201-212 (1995).
5. Prieto-Diaz, R., Implementing faceted classification for software reuse. In: 12th Intemational Conference on Software Engineering, pp. 300-304. (1990).
6. IEEE, IEEE standard for information technology—software reuse—data model for reuse library interoperability: basic interoperability data model (BIDM), IEEE Std 1420.1-1995. (1995).
7. Happel, H.J., Korthaus, A., Seedorf, S., and Tomczyk, P., KOntoR: an ontology-enabled approach to software reuse, In: Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2006), pp. 349-354 (2006).
8. Alnusair, A. and Zhao, T. Component search and reuse: an ontology-based approach, In: Proceedings of the IEEE International Conference on Information Reuse and Integration, (IRI 2010), pp.258-261. (2010).

9. Nianfang, X, Xiaohui,Y. and Xinke, L. Software components description based on ontology. In: 2010 Second International Conference on Computer Modeling and Simulation. pp.423-426. (2010).
10. Peng, Y., Peng, C., Huang, J. and Huang, K. An ontology-driven paradigm for component representation and retrieval. In: IEEE Ninth International Conference on Computer and Information Technology. pp.187-192. (2009).
11. Alnusair, A., Rawashdeh, M., Alhamid, M.F., Hossain, M.A., and Muhammad, G.: Reusing software libraries using semantic graphs. In: 17th International Conference on Information Reuse and Integration Proceedings, pp. 531-540, IEEE, U.S.A. (2016).
12. Akar, Z., Halaç, T. G., Ekinci, E. E. , and Dikenelli, O. Querying the web of interlinked datasets using VOID descriptions. *LDOW* 2012, (2012).
13. Weide, Bruce W., William F. Ogden, and Stuart H. Zweben. Reusable software components. Advances in computers 33, 1-65. (1991).
14. The Apache Software Foundation, Welcome to Apache Maven, https://maven.apache.org/index.html, son erişim 07.09.2017.
15. The Apache Software Foundation, Introduction to the POM, https://maven.apache.org/guides/introduction/introduction-to-the-pom.html, son erişim 07.09.2017
16. The Apache Software Foundation, What is maven, https://maven.apache.org/what-is-maven.html, son erişim 11.09.2017