Yazılım Geliştirmede Model Dönüşümü ve Model Dönüşüm Dilleri

Arda GÖKNİL[1] N. Yasemin TOPALOĞLU[2]

[1,2] Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova-İzmir

[1]e-posta: arda.goknil@ege.edu.tr [2]e-posta: yasemin.topaloglu@ege.edu.tr

Özet

Akademisyenler ve endüstri temsilcilerinden oluşan uluslararası bir kuruluş olan *Object Management Group* (OMG) tarafından tanımlanan *Model Tabanlı Mimari* (Model Driven Architecture- MDA) yazılım geliştiriminde model merkezli yaklaşımların kullanılmasını önermektedir. Bu yaklaşımın ana hedefi, yazılım geliştirme işlemini platform bağımsız seviyede tutmak ve platform bağımsız modellerden platform bağımlı modellere geçişin otomatik model dönüşümleriyle gerçekleştirilmesidir. Bu amaçla model dönüşümlerinin belirli bir standarda oturtulabilmesi için OMG tarafından bir çalışma yürütülmektedir. Bunun bir ürünü olan *Query/View/Transformation (QVT)* dokümanında ortaya konulmaya başlanan standartlar, model dönüşümlerinin gerçekleştirilmesi için model dönüşüm dillerinin geliştirilmesi zorunluluğuna işaret etmektedirler. Bu bildiride bir model dönüşüm dilinden istenen özellikler ve yürütülen çalışmalar kısaca tanıtılarak, önerilen bir model dönüşüm dili tanıtılmaktadır.

Abstract

Model-Driven Architecture (MDA) which is defined by the Object Management Group (OMG) advocates to use model driven approaches in software development. OMG is an international organization formed of academicians and industry representatives. The main philosophy of MDA is to develop software as platform independent models and then transform these models into platform specific models by using model transformation tools. OMG has an ongoing project to specify the standards in model transformations. The Query/View/Transformation document that is a result of this project defines several standards which focus on the need of developing model transformation languages. In this paper, we present the model transformation language standards and discuss the related work in this field. Also we introduce the basic characteristics of a model transformation language that we propose

1. Giriş

Yazılım endüstrisi temsilcilerinin ve akademisyenlerinin oluşturduğu uluslararası bir kuruluş olan *Object Managament Group* (OMG), 2001 yılında model tabanlı yazılım geliştirmeyi destekleyen bir dizi standart oluşturulması için Model Tabanlı Mimari'yi (*Model Driven Architecture*) [9] tanıtmıştır.

Model Tabanlı Mimari (MTM) [9], yazılım sistemlerinin geliştirilmesinde soyutlama düzeyini artırmak için model merkezli yaklaşımların kullanılmasını önermektedir. Model tabanlı mimarinin ana amacı, yazılım geliştirme işlemini platform bağımsız seviyede tutarak kolaylaştırmaktır. Yazılım sistemlerinin geliştirilmesinde nesne modellerinin çalıştırılabilir bileşenlere ve uygulamalara dönüştürülmesini öneren bu yaklaşımda farklı soyutlama seviyesindeki modeller

arasındaki dönüşümler büyük önem taşımaktadır. *MTM* model tanımında üç soyutlama seviyesi tanımlanmıştır:

- Platform bağımsız modeller (PBM),
- Platforma özgü modeller (POM),
- Kaynak kod.

Farklı soyutlama seviyeleri arasındaki geçişlerin model dönüşümleri aracılığıyla gerçekleştirilmesi amaçlanmaktadır. Bir model dönüşümü, değiştirilmek istenen modeli kaynak model olarak alır ve bir dizi dönüşüm kuralını uygulayarak bir ya da daha fazla hedef model haline çevirir [3]. MTM'nin tanımlanmasından sonra modellerin dönüşümüyle ilgili çeşitli teknikler önerilmiştir. OMG tarafından model dönüşümlerinin belirli bir standarda oturtulabilmesi için bir çalışma yürütülmektedir. Bu çalışma kapsamında ortaya konulan *Query/View/Transformation* [11] dokümanında belirtilen standartlar model dönüşümlerinin gerçekleştirilmesi için model dönüşüm dillerinin geliştirilmesi gerektiğini belirtmektedir.

Bu çalışmada, MTM kapsamında yapılan tanımlamalara göre bir model dönüşüm dilinden beklenen özellikler incelenerek, bu konuda yürütülen çalışmalar kısaca tanıtılmış ve önerilen bir model dönüşüm dilinin genel özellikleri açıklanmıştır.

Bölüm 2'de model dönüşüm dillerinin MTM yaklaşımındaki yeri incelenmiştir. Bölüm 3'te Model Tabanlı mimari içerisinde yer alan genel kavramlar açıklanmış ve Bölüm 4'te genel olarak QVT tarafından belirtilen istekleri karşılayabilecek bir model dönüşüm dilinin özellikleri belirtilmiştir. Bölüm 5'te MTM kapsamında literatürde geliştirilen model dönüşüm dili çalışmaları tartışılmış ve Bölüm 6'da rol tabanlı bir model dönüşüm dilinin genel özellikleri tanıtılmıştır. Yedinci bölüm sonuç kısmını içermektedir.

2. Model Tabanlı Mimariye Genel Bir Bakış

Model Tabanlı Mimarinin merkez noktası, geliştiriciler tarafından alana özgü kavramlarla yani kullanılan teknolojik platformdan bağımsız olarak tanımlanan modellerin, kod oluşturulmasında kullanılabilecek platform bağımlı modellere dönüştürülmesi, yani model dönüşümleridir. Sendall ve Kozaczynski [14] model dönüşümlerini, bir veya daha fazla kaynak modeli girdi olarak alıp bir dizi dönüşüm kuralını uygulayarak sonuçta bir veya daha fazla hedef modeli oluşturma süreci olarak tanımlamakta ve bu dönüşümün gerçekleşebilmesi için, kaynak ve hedef modellerin hem soyut sözdizimin hem de anlamının (*semantics*) anlaşılmasına gereksinim olduğunu belirtmektedirler.

Modeller arası işbirliğini sağlamada önemli bir bileşen, üst modelleme (*meta-modelling*) kavramı ve üst modellerdir. Bir üst model (*meta-model*), belirli bir modelleme dilinde geçerli olan modelleri tanımlayan bir modeldir [13]. Üst model gibi üst düzey tanımlamalar, modeller üzerinde çalışılmasını kolaylaştırırlar [1].

OMG tarafından 1997 yılında tanımlanan *Unified Modeling Language* (UML), son yıllarda yazılım dünyasında yaygın olarak kullanılan bir modelleme standardı olarak yer almaktadır. UML modelleri ile, geliştirilen sistemin yapısı ve davranış şekli, farklı bakış açılarından gösterilebilir. Bunun yanı sıra, UML modellerinin metinsel olarak açıklanması için *Object Constraint Language* (OCL) [17] tanımlanmıştır.

Üst model kavramı, UML tanımının [8] da temelini oluşturmaktadır. UML üst modeli, geçerli olan UML modellerinin sağlaması gereken kuralları tanımlamaktadır. OMG tarafından, sadece UML üst

modelini değil, benzer dillerin de tanımını sağlamak için *Meta-Object Facility* (MOF) [10] adı verilen bir çerçeve tanımlanmıştır. UML üst modeli, nesneye yönelik yazılım sistemleri için model oluşturmasında kullanılan dili tanımlarken, MOF, üst modeller oluşturmak için kullanılan dili tanımlar [10].

3. Model Dönüşüm Yaklaşımları

MTM kapsamında model dönüşümü için farklı yaklaşımlar tanımlanmıştır. Gerçekleştirimler farklılık göstermekle birlikte model dönüşümleri, temelde uygulama modellerinin kaynak ve hedef desenler doğrultusunda dönüşüm kuralları uygulanarak dönüşüme uğratılmasını sağlamaktadır.
Sendall ve Kozaczynski [14], model dönüşümlerinin tanımlanmasında araçlar tarafından kullanılan yaklaşımları üçe ayırmışlardır:

- Doğrudan Model İşleme: Bu yaklaşımda amaç, *Rational* [12] gibi yazılım geliştirme araçları ile model üzerinde işlem yapılabilmesini sağlayan API'ler yardımıyla modellerin dönüşüme uğratılmasıdır. Kullanılan API'ler ya da *Script* dilleri araç tarafından sunulduğu için başka bir araç kullanılması durumunda geçerliliklerini kaybederler.
- Ara Gösterim: Ara gösterim yaklaşımında ise gerçekleştirim, modeli standart bir biçimde tanımlar ve bu tanım üzerinde çalışır. Örneğin uygulama modelleri, kaynak ve hedef desenleri XMI formatında [16] tutulur. Her biri üç ayrı XMI dosyasında tutulan uygulama modeli, kaynak ve hedef desen belleğe yüklenir. Daha sonra kaynak desen, uygulama modeli içerisinde aranarak uygulama modelinin dönüşüme uğratılacak kısımları belirlenir ve kaynak ve hedef desene göre ilgili kural dosyası işletilir. Genel olarak bu yaklaşımda dönüşüm kurallarının XSLT formatında kodlanması tercih edilmektedir. XSLT [18] formatında kodlanmış kurallar belleğe yüklenip uygulama modeli üzerinde işletilerek dönüşüm gerçekleştirilmiş olur. Model dönüşümünün doğru gerçekleşip gerçekleşmediği ise ilgili hedef desenlerin dönüşüme uğrayan uygulama modeli üzerinde aranması ile belirlenir. Bu yaklaşımın bazı dezavantajları bulunmaktadır. Örneğin kaynak ve hedef desenlerin statik olarak XMI dosyaları içerisinde kodlanması desenlerdeki her bir varyasyon için değişik XMI dosyalarının var olmasını gerektirmektedir. Bu nedenle aynı dosya içerisinde *And, Or, XOR* gibi mantıksal operatörler belirtilememektedir. Yine ayrıca kuralların da statik olarak kodlanması koşulların kontrol edilerek kuralların işletilmesini olanaksız kılmaktadır.
- Dönüşüm dili desteği: Bu yaklaşımda ise dönüşümlerin tanımlanması ve uygulanması için tanımlanan dil, programlama dillerinde olduğu gibi kontrol ve döngü ifadeleri içerebilir. Dönüşüm dili desteği ile ara gösterimde yapılamayan kaynak ve hedef desenlerdeki değişken noktaların yakalanması, kuralların esnek olarak işletilmesi gibi işlemler gerçekleştirilir. Dönüşüm dilleri bu desteği *And, Or, XOR, If* gibi mantıksal ifadeleri kullanarak verir.

4. Model Dönüşüm Dili Standartları

Model dönüşümlerinde belirli bir standardın sağlanması için, OMG tarafından *Query/View/Transformation* [11] adı verilen bir çalışma yürütülmektedir. Bu çalışmada, model dönüşüm dillerinin sağlaması gereken aşağıdaki standartlar tanımlanmıştır [11]:

- MOF modellerini sorgulayacak bir dilin tanımlanması
- Dönüşüm tanımları için bir dilin tanımlanması
- Bir modelin değişik durumlarının ortaya konulması

- Dönüşümlerin eksiksiz şekilde gerçekleştirilebilmesi için önerilen dilin tanımlayıcı (*declerative*) olması
- Önerilen dillerin, MOF modelleri şeklinde ifade edilmesi.

Bu standartların karşılanması için gerekenler aşağıda özetlenmiştir [11]:

- Dönüşüm kuralları, model elemanlarını tek tek işleyebileceği gibi, model elemanları kümesini de işleyebilmelidir.
- Kurallar, kaynak ve hedef model elemanları arasında ilişkiler kurarak işletilmelidir.
- Kurallar, değişik üst seviyelerde eşleme yapabilmeli ve model elemanı oluşturabilmelidir. Bunun içinde model dönüşümlerinde dinamik tipleme (*dynamic typing*) desteklenmelidir.
- Dönüşümler, hem kaynak modelde hem de hedef modelde oluşabilecek değişkenlikleri işleyebilmelidir.
- Çeşitli hedef elemanları tek bir kural içerisinde tanımlanabilmelidir.
- Tek bir hedef elemanı, çeşitli kurallar tarafından tanımlanabilmelidir. Böylece değişik kurallar aynı nesne için değerler üretebilir.
- Okunabilirliği ve modülerliği sağlamak için kurallar gruplandırılmalıdır.
- Model dönüşüm dili ya da dili işleyen dönüşüm motoru, dönüşüm kurallarının bir sıra dahilinde işletilmesini desteklemelidir.

5. Model Dönüşüm Dili Yaklaşımları

5.1. Grafiksel Model Dönüşüm Dili: MOLA

MOLA [4] dili ile kontrol yapıları gibi geleneksel programlama dili kavramları model dönüşüm dilleri içerisinde uygulanmaya çalışılmıştır. MOLA'da görsel dil elemanları ile metinsel kısıt dili birleştirilmiştir. Dönüşümlerin görsel olarak ifade edilmesi, kaynak ve hedef modeller arasındaki eşlemelerin doğrudan tanımlanabilmesi açısından önemlidir. Bu özellik MOLA gibi görsel dönüşüm dillerinin oluşturulmasının ana hedefidir. Bu dönüşüm dillerinin temellerini çizge dönüşümleri (*graph transformations*) oluşturmaktadır.

MOLA'da model elemanları arasındaki ilişkiler, dönüşüm kuralları, "*IF-Then-Else*" gibi koşul elemanları görsel olarak ifade edilmektedir. Bu "*IF-Then-Else*" yapısı ile dönüşümler hem kaynak modelde hem de hedef modelde oluşabilecek değişkenlikleri işleyebilmektedir. Dil içerisindeki anahtar eleman, grafiksel döngü yapısıdır. Döngü yapısı ile aynı özellikte birden fazla model elemanı işlenebilmekte ve üzerinde dönüşüm kuralları uygulanabilmektedir. Döngü yapısı sayesinde dönüşüm kuralları, model elemanlarını tek tek işleyebildiği gibi model elemanları kümesi halinde de işleyebilmektedir. Model elemanlarının taşıdığı kısıtlayıcı özellikler ise metinsel kısıt dili ile ifade edilmektedir.

MOLA içerisinde tanımlanan kurallar, aynı zamanda izlenebilirliği de sağlamaktadır. Örneğin bir dönüşüm içerisinde kaynak ve hedef model elemanı arasında bir kural tanımlanabilir ve başka bir dönüşüm hedef modelde yer alan model elemanına bu kural üzerinden erişebilir. Buna karşın MOLA, dönüşüm kurallarının bir sıra dahilinde işletilmesini desteklememektedir. Bu dil, dönüşüm kurallarının hangi sıra dahilinde işletileceği sorumluluğunu geliştiriciye bırakmaktadır. Geliştiricinin kurallar arasındaki bütün ilişkileri bilmesi ve kaynak model üzerinde tanımlanan bütün dönüşüm kurallarının işletilme sırasını, bu ilişkileri göz önüne alarak tanımlaması gereklidir.

5.2. Tanımlayıcı Bir Model Dönüşüm Dili

Bu dönüşüm dili, MOF meta elemanlarını da kapsayan bir dönüşüm üst modelini (transformation metamodel) içermektedir [2]. Bu dönüşüm üst modeli, dil içerisinde tanımlı bütün kural ve dönüşüm elemanlarını kapsamaktadır. Dil kendi içerisinde, dönüşüm deseni ve kurallar olmak üzere ikiye ayrılmaktadır. Dönüşüm deseni, uygulama modelinde değiştirilecek model elemanlarını tespit eden sorgu cümleciklerini oluştururken, kurallar ise uygulama modeli üzerinde çalıştırılan dönüşüm işlemlerini temsil etmektedir.
Örnekte, bu dil kapsamında UML sınıfı ve o UML sınıfının kendi içerisinde yer alan ya da üst sınıflarından aldığı *attribute'*leri tanımlayan bir desen tanımı verilmektedir [2].

```
PATTERN hasAttr(C, A)
FORALL Class C, Attribute A, Class C2
WHERE A.owner = C
     OR (C.super = C2 AND hasAttr(C2, A))
```

Bu desen tanımları daha sonra kural tanımları içerisinde kullanılmaktadır. Desen tanımları ile MOF modellerini sorgulayacak bir dil tanımlanırken, desen tanımlarının kural tanımları ile birlikte kullanılması ile dönüşüm işlemleri için bir dil oluşturulmuştur. Kural tanımı ise şu şekilde yapılmaktadır:

```
RULE Uclass2Jintf(Cls)
FORALL Class Cls
MAKE Interface Intf
LINKING Cls to Intf by Uclass2Jintf
```

Uclass2Jintf kuralı *Cls* adlı sınıfı parametre olarak almaktadır ve bu sınıfı *interface* olarak değiştirmektedir. *Make* satırı ile işlem gerçekleştirilir. "FORALL Class Cls" ibaresi ise bu işlemi bütün *Cls* için uygula anlamına gelmektedir. Dil içerisinde kural tanımlarında hangi kaynak elemanlarının hangi hedef elemanları ile eşleştirildiğini göstermek için *Tracking* ifadeler konmaktadır. Bu ifadeler ile kural içerisinde kaynak ve hedef elemanları ilişkilendirilerek izleme yapılabilir.
Linking deyimi ile kaynak modeldeki *Cls* model elemanının hedef modeldeki *Intf* elemanı ile kural tanımı içerisinde ilişkilendirildiği belirtilmektedir. *Linking* deyimi ile aslında QVT dökümanında yer alan model kaynak ve hedef model elemanları arasında izlenebilirlik (*treceabiality*) sağlanmaktadır. Bu deyim sayesinde diğer kurallar içerisinde *Uclass2Jintf* kuralının sonucunda oluşan model elemanına yani *Interface*'e erişilebilmektedir.
Dil, dönüşüm kurallarının bir sıra halinde işletilmesini içsel olarak desteklemektedir. Dönüşümü geliştiren kişi, laynak ve hedef model elemanları arasında dönüşüm kurallarını tanımladıktan sonra dönüşümü işleyen motor, bu kurallar arasındaki ilişkilerden yola çıkarak kuralların işletim sırasını kendisi çalışma anında belirlemektedir.

5.3. Birleştirilmiş Model Dönüşüm Dili Yaklaşımı

Bu çalışmada, QVT yaklaşımının katmanlar arası dönüşümü kısıtlaması nedeniyle, bu dönüşümleri de destekleyen ortak bir dönüşüm üst modelinin oluşturulması önerilmektedir [6]. Bu üst model, MOF model elemanlarından bağımsız bir üst model olacaktır ve dönüşüm işlemleri, bu ortak üst model üzerinde gerçekleşecektir. Ortak bir üst model kullanılması, diğer üst modellerin bu üst

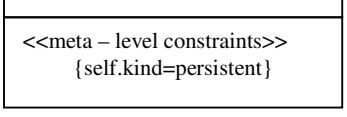
modele eşlenmesini gerektirmektedir. Bu nedenle MOF üst modeliyle tanımlanan üst model arasında *instanceOf* ilişkileri oluşturulmuştur. Her bir model elemanı bu *instanceOf* ilişkisi ile tanımlanmaktadır.
Bu çalışmada iki problem incelenmiştir. Bu problemlerden birincisi MOF'nin dört katmanlı modelleme mimarisinde M1 seviyesi ile M0 seviyesi arasındaki ilişkiyi tanımlamamasıdır. Kurtev ve Berg [6], bu mimarinin dört katmanlı olmayıp, üç katmanlı olduğunu savunmaktadır. Çünkü, MOF tarafında M1 seviyesindeki hangi model elemanlarının örneklenebilir olduğu ve M0'da hangi elemanlarla temsil edildiği bilgisi bulunmamaktadır. İkinci problemde, ilk problemle bağlantılı olarak M1 model elemanlarıyla M0 model elemanları arasında türetme ilişkisi MOF tarafından tanımlanan *instanceOf* ilişkisi ile açıklanmadığı ileri sürülmektedir. Bu nedenle değişik seviyeler arasındaki model dönüşümlerini destekleyecek bir model dönüşüm dili bu seviyeler arasındaki farklı *instanceOf* ilişkilerini de desteklemek durumundadır.

6. Rol Tabanlı Model Dönüşüm Dili Önerisi

Bu bölümde, model dönüşümlerinin gerçekleştirilmesi için üzerinde çalışmakta olduğumuz yaklaşım hakkında bilgi verilecektir. Bir model dönüşüm dilinin tanımlanmasında ve gösterilmesinde rol kavramından yararlanılabilir. Burada rol, model içerisinde yer alan ve aynı özellikleri taşıyan sınıfların model içerisindeki yapısal ve davranışsal özelliklerini tanımlamaktadır. Rol tabanlı model dönüşüm diliyle önerilen, dönüşüm dili içerisinde tanımlı bütün yapıların roller ile ifade edilmesidir. Bunu sağlayabilmek için dil içerisinde tanımlanacak roller, hem kaynak üst modeli, hem hedef üst modeli ve hem de dönüşüm dilinin üst modeli desteklemelidir. Örneğin, kaynak ve hedef desenler içerisinde yer alan model elemanlarının ifade edilmesi model elemanlarıyla ifade edilirken dil içerisinde yer alan dönüşüm kuralları, alt program çağrımları, kontrol yapıları rol elemanlarıyla ifade edilmelidir. Bu yaklaşımda şöyle bir benzetmede bulunulabilir: nesneye yönelik programlama dillerinde nasıl dil içerisindeki her eleman bir sınıf ise rol tabanlı bir model dönüşüm dilinde, dildeki her eleman bir roldür.

6.1. Model Dönüşüm Dili İçerisindeki Yapılar

Rol tabanlı model dönüşüm dilinde rollerin ve dönüşümlerin ifade edilmesinde gösterim olarak *MetaRole-Based Modelling Language* (RBML) [5] dilinden yararlanılmıştır. Bu gösterim, rolü oynayan sınıfların sayılarını belirtmemize ve bu sınıflara ait kısıtlamaları üst (meta) seviyede OCL [17] kullanarak göstermemize olanak sağlamaktadır.
Bu yaklaşım, model dönüşümlerini görsel olarak ifade edebilecek ve OCL gibi metin tabanlı kısıt dilini, bu görselliğin içerisine katarak dönüşümlerin daha yapısal olarak ifade edilmesini sağlayabilecektir. Örneğin, bir UML modelinin veri tabanı dönüşümlerinde kalıcı (*persistent*) sınıflar veri tabanında tablolara karşılık gelecek şekilde dönüştürülmek istenir. Bu amaçla kaynak model için tanımlanan rol model Şekil-1'deki gibi tanımlanabilir:



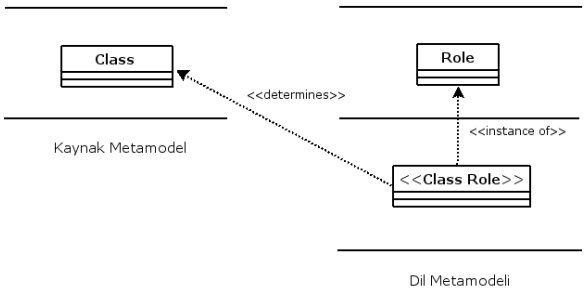
Şekil 1. Sınıf Model Elemanlarının Rol ile Gösterilmesi.

<<Class Role>> doğrudan kaynak meta modeldeki sınıfa karşılık gelmektedir ve bu rol model ile model içerisinde o sınıftan türeyen model elemanlarının kalıcı olanları ifade etmektedir. **0..*** ise bu rolü oynayan birden fazla sınıf olabileceği belirtilmektedir. "*" sembolü, aslında o rolü oynayan sınıfların bir döngü ile işleneceğini belirtmektedir. Bu grafiksel rol gösterimleriyle birlikte OCL tabanlı metinsel kısıtlamaların kullanılması, dönüşüm için gerekli bütün kısıtların ve kuralların gösterilmesine olanak sağlamaktadır.

Her bir rol model, üst (meta) seviyede bulunan bir üst sınıftan türetilmektedir. Bu üst sınıflar ise kendilerinden türeyen model elemanlarının aynı zamanda tipini belirtmektedir. Şekil-1'de rolü oynayan model elemanlarının tipi *Class*'dır. <<Class Role>>, o rolü oynayan model elemanlarının tipini belirtilmekte ve belirtilen tipe göre model elemanları eşlenebilmektedir. Rol gösteriminde tanımlanan bütün özellikler ve kısıtlamalar, o rolü oynayan model elemanlarını eşleyebilmek için kullanılmaktadır. QVT kapsamında geliştirilen diğer dillerde, model elemanları arasına izlenebilirlik linkleri (*treceability link*) konulurken, rol model tabanlı bir dilde, rol modeller arasına konulan dönüşüm kurallarını temsil eden roller tanımlanarak çözülebilir. Kaynak ve hedef model elemanlarının izlenmesi bu roller üzerinden gerçekleştirilir. Dil içerisinde birden fazla adımdan oluşan dönüşümler tanımlanırken işlem, alt dönüşümlere bölünebilir ve alt dönüşümler de bir dönüşüm oldukları için ana dönüşüm içerisinden çağrılabilir.

6.2. Model Dönüşüm Dili Elemanları İle Üst model Elemanları Arasındaki ilişki

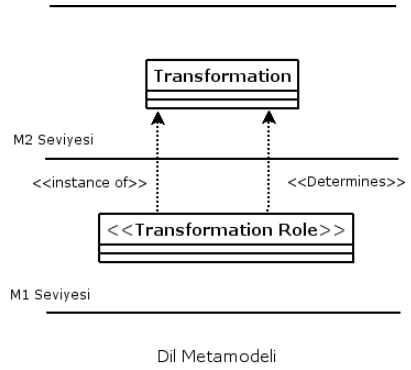
QVT, model dönüşüm dillerinin MOF modelleri şeklinde ifade edilmesini tanımlamaktadır. Bu nedenle model dönüşüm dilleri temelde üst modeller ile gösterilmektedir. Dönüşüm dilinde rol model kullanmanın sonucu olarak, üst model içerisinde tanımlanan bütün dil yapılarına o üst sınıfları esas alan roller aracılığıyla erişilebilecektir. Bu da bize dil içerisindeki bütün tanımların roller aracılığıyla gerçekleştirilebileceğini, yani tamamıyla rol tabanlı bir model dönüşüm dili geliştirilebileceğini göstermektedir. Dil içerisindeki bütün kavramların roller ile ifade edilmesine karşılık, dile üst modeli ile olan ilişkilere bakıldığında iki değişik rol yer almaktadır. Birinci rolde, dil içerisinde yer alan rolü oynayan sınıflar kaynak ya da hedef modellerde yer almaktadır. Bu nedenle rolün tanımladığı üst sınıflar kaynak ya da hedef üst modellerinde yer alırken rolün türediği üst sınıf dil üst modelinde yer almaktadır.



Şekil 2. Kaynak Üst Modeli - Dil Üst Modeli İlişkisi.

Şekil-2'de kaynak üst modeli ile dil üst modeli arasındaki ilişki görülmektedir. Dil üst modelinde yer alan *Role* üst sınıfından örneklenen <<Class Role>> kaynak üst model içerisinde yer alan <<Class>> üst sınıfını tanımlamakta ve bu üst sınıftan örneklenen model elemanları bu <<Class

Role>> rolünü oynamaktadır. Kaynak ya da hedef modellerde yer alan model elemanlarının oynadığı roller aynı zamanda dil üst modelinde yer alan *Role* üst sınıfından türetilmektedir.



Şekil 3. Dil Üst Modelinde Rol – Üst Sınıf İlişkisi.

Şekil-3, ikinci tip rollerin dil üst modelindeki üst sınıflarla olan ilişkisini göstermektedir. Dil içerisinde yer alan ikinci tip roller, dil içerisinde dönüşüm kuralları gibi dil üst sınıfında yer alan üst sınıflar tarafından oynanmaktadır. Bu tip sınıflar dil içerisindeki temel kavramları oluşturmaktadır. Örneğin dönüşüm bilgisini tutan *Transformation* sınıfı, dönüşüm kurallarını temsil eden *TRule* sınıfı gibi üst sınıfların oynadığı rollerin dil üst modeliyle olan ilişkileri daha farklıdır. Dönüşümün kendisini tanımlayan <<Transformation Role>> rolü, Şekil-2'de olduğu gibi kaynak ya da hedef model elemanları tarafından oynanmayıp, doğrudan doğruya dil üst modeli yer alan *Transformation* üst sınıfı tarafından oynanmaktadır. Bunun gibi dil üst sınıfları tarafından oynanan roller Şekil-2'deki gibi *Role* üst sınıfından örneklenmeyip yine rolü oynayan üst sınıftan türetilmektedir. Şekil-3'teki <<Transformation Role>> rolü ile *Transformation* üst sınıfı arasında hem <<determines>> hem de <<instanceOf>> ilişkisi mevcuttur. <<Transformation Role>> hem *Transformation* üst sınıfından türediği gibi hem de bu üst sınıfı tanımlamaktadır.

7. Sonuç

Model dönüşümlerini gerçekleştirebilmek için önerilen değişik yaklaşımlar içerisinde uygulanabilirliği en yüksek olan yaklaşım, model dönüşüm dillerinin kullanılmasıdır. Model dönüşüm dillerinin, mantıksal işlemleri desteklemesi, ve kaynak, hedef desen tanımları hem de dönüşüm kurallarının her üçünü birden tek bir çatı içerisinde tanımlayabilmesi, kendisini diğer yaklaşımlardan bir adım öne çıkarmaktadır.

Rol tabanlı model dönüşüm dili ile dönüşüm dili içerisindeki tüm yapıların roller ile ifade edildiği ve aynı seviyede ve farklı seviyelerde dönüşümlerin gerçekleştirilebildiği bir dönüşüm dili hedeflenmektedir. Bu dilin diğer dillerden temel farkı, görsel gösterim ile metinsel kısıt dilini birleştirebilmesi ve model iyileştirmesi gibi (model refactoring) karmaşık dönüşüm işlemlerini karşılayabilecek yapıda olmasıdır.

Kaynakça

[1]. Atkinson, C., Kühne, T.: "The Role of Metamodeling in MDA" Proceedings of the Workshop in Software Model Engineering (WISME@UML'2002), 2002

[2]. Duddy, K., Gerber, A., Lawley, M., Raymond, K., Steel, J.: Model Transformation: A declarative, reusable patterns approach. In Proceedings 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003), pp 174-185

[3]. Judson, S., France, R., Carver, D.: Specifying Model Transformations at the Meta-model Level. Workshop in Software Model Engineering (WISME@UML'2003), San Francisco USA, October (2003)

[4]. Kalnins, A., Barzdins, J., Celms, E.: Model Transformation Language MOLA. Model Driven Architecture: Foundations and Applications (MDAFA 2004), Sweden, June (2004)

[5]. Kim, D., France, R., Ghosh, S., Song, E,: Using Role-Based Modeling Language (RBML) as Precise Characterizations of Model Families. The 8th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), (2002).

[6]. Kurtev, I., Berg, K.: A Language for Model Transformations in the MOF Meta-modeling Architecture. Model Driven Architecture: Foundations and Applications (MDAFA 2004), Sweden, June (2004)

[7]. Kurtev, I., Berg, K.: Unifying Approach for Model Transformations in the MOF Metamodeling Architecture. 1st European Workshop on Model-Driven Architecture with Emphasis on Industrial Applications (MDA-IA 2004), Enschede The Netherlands, March (2004)

[8]. OMG, UML Infrastructure Specification Version 2.0, The Object Management Group, www.omg.org

[9]. OMG: MDA Guide Version 1.0.1. The Object Management Group, Document Number: omg/2003-06-01 (2003)

[10]. OMG, Meta Object Facility (MOF) Specification, The Object Management Group, 2000, www.omg.org

[11]. OMG: MOF 2.0 Query/Views/Transformations RFP. The Object Management Group (2002)

[12]. Rational XDE, http://www-306.ibm.com/software/awdtools/developer/rosexde/

[13]. Seidewitz, E.: "What Models Mean" IEEE Software, Vol. 20, pp. 26-32, 2003

[14]. Sendall, S., Kozaczynski, W.: Model Transformation – the Heart and Soul of Model-Driven Software Development. IEEE Software Sep/Oct. (2003), pp. 42-45

[15]. Staron, M., Kuzniarz, L.: Implementing UML model transformations for MDA. 11th Nordic Workshop on Programming and Software Development Tools and Techniques (NWPER'2004), Turku Finland, August (2004)

[16]. Wagner, A.: A pragmatical approach to rule-based transformations within UML using XMI.difference. Workshop on Integration and Transformation of UML models (WITUML 2002), Malaga Spain, June (2002)

[17]. Warmer, J., Kleppe, A.: Object Constraint Language, The: Getting Your Models Ready for MDA. Addison-Wesley Publishing, 2003

[18]. XSL Transformations, http://www.w3.org/TR/xslt