Yazılım Ürün Hatlarında Yetenek Modeli ve Üstmodel Senkronizasyonu için bir Yöntem

Ferhat Erata[1,2], Moharram Challenger[2], Serhat Gezgin[1], Akgün Demirbaş[1,2], Mehmet Önat[3] ve Geylani Kardaş[1,2]

[1] UNIT Bilişim Teknolojileri Ar-Ge Ltd., Ar-Ge Bölümü, Urla, İzmir
{ferhat.erata, serhat.gezgin, akgun.demirbas}@unitbilisim.com
[2] Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, Bornova, İzmir
{geylani.kardas, moharram.challenger}@ege.edu.tr
[3] KoçSistem Bilgi ve İletişim Hizmetleri A.Ş., Ar-Ge Merkezi, Üsküdar, İstanbul
mehmet.onat@kocsistem.com.tr

Özet. Yazılım ürün ailesi içerisindeki ortaklıklar ve değişkenlikler genel olarak yetenek modelleri ya da alana özgü diller ile ifade edilirken, ürün ailesi için oluşturulan yazılım platformları model güdümlü yazılım geliştirme teknikleri kullanılarak oluşturulabilmektedir ve temelde çeşitli soyut sözdizimi tanımlamalarına dayanmaktadır. Bu bildiri, model güdümlü ürün hattı yaklaşımının alan mühendisliği sürecinde problem uzayında bulunan ortaklıkların ve değişkenliklerin modellenmesi için kullanılan yetenek modelleri veya alana özgü dil modelleri ile yazılım ürün mimarisi arasında izlenebilirliği arttıran ve yazılım mimarisinin artımlı geliştirilmesini olanaklı kılan bir senkronizasyon yöntemi önermektedir. Yöntemin SAT ve CSP çözücüleri ile entegrasyonu sayesinde alana özgü dilin doğrulama ve sınanması için de kullanılabilmektedir. Bu yönteme dayalı olarak geliştirilen açık kaynak kodlu araç yardımıyla yöntem, bir endüstriyel ar-ge projesi kapsamında geliştirilen yazılım ürün hattı mimarisinde başarıyla uygulanmış ve kullanımı bu bildiride gösterilmiştir.

Anahtar Kelimeler: Alana Özgü Diller, Alloy, Clafer, Değişkenlik Yönetimi, ECore, Model Dönüşümü, Yazılım Mimarisi, Yazılım Ürün Hattı Mühendisliği.

1 Giriş

Yazılım ürün hattı mühendisliği süreçlerinde, yeniden kullanılabilir bir platform ve ürün hattı mimarisinin geliştirildiği bir alan mühendisliği ile alan mühendisliği süreci sonucunda elde edilen varlıklardan ürünlerin geliştirildiği bir uygulama mühendisliği süreci yer alır [5]. Alan mühendisliği sürecinde alan içerisindeki ortaklıklar ve değişkenlikler belirlenmekte ve genellikle bir yetenek modeli ile ifade edilmektedir. Yetenek bağımlılıkları ve kısıtları (İng. *dependencies and constraints*) dikkate alınarak bu yeteneklerin ihtiyaca göre bir alt kümesi seçilir ve seçilen yeteneklere sahip yeni bir yazılım elde edilir. Bu süreçte, Model Güdümlü Mühendislik (İng. *Model Driven Engineering*) teknikleri sıklıkla kullanılmakta ve özellikle Alana Özgü Diller (İng. *Domain Specific Languages*) (DSL) yetenek modelleri gibi, yazılım ürün hatlarında

problem uzayını tanımlamada kullanılmaktadır [3]. DSL geliştirmede yaygın olarak kullanılan, açık kaynak kodlu ve gelişmiş çatılardan biri Eclipse EMF (İng. *Eclipse Modeling Framework*) teknolojisidir [7]. Bir tür DSL olan yetenek modelleme dillerini, gelişmiş bağımlılık ve kısıt yönetimi açısından değerlendirdiğimizde ticari olanları Pure::Variants [20] ve Gears [21], açık kaynak kodlu olanları ise TVL[8] ve Clafer[1] dilleri oluşturmaktadır [9]. Bunların arasında Clafer sağladığı çoklu örnekleme (İng. *multiple instantiation*), ilişkisel referans kurma ve kalıtım desteği ile öne çıkmaktadır. Ancak Clafer aracının Eclipse EMF teknolojileri ile entegrasyonu yoktur ve Eclipse Modelleme alt yapısı ile geliştirilmemiştir. Bu sebeple bu araç ile geliştirilen yeteneklerin EMF tabanlı alana özgü dil geliştirme aktivitelerinden biri olan üst modellemeye geçişte izlenebilirliği kaybolmaktadır. Bu ihtiyaçlardan yola çıkarak bu bildiride tanıtılan yöntem, yetenek modelleri ve konfigürasyonları ile üstmodel ve modelleri arasında iki yönlü dönüşümü otomatik olarak gerçekleştirebilmektedir. Yöntemin SAT (İng. *Boolean Satisfiability Problem*) ve CSP (İng. *Constraint Satisfaction Problem*) çözücüleri ile entegrasyonu sayesinde alana özgü dilin doğrulama ve sınanması da mümkündür. Araç, Eclipse EMF tabanlı model güdümlü geliştirme yöntemleri kullanılan tüm projelerde rahatlıkla kullanılabilecek düzeydedir.

Bildirinin takip eden ikinci bölümünde, yöntemimize uygun yetenek modelleme dilleri ve alana özgü dil geliştirme teknikleri anlatılmıştır. Sonraki bölümde ise yapılan çalışmanın mimari bileşenleri ve yöntemi tanıtılmış, Clafer ve ECore dilleri arasındaki dönüşümlerin detayları aktarılmıştır. Dördüncü bölümde önerilen yöntemin gerçek bir endüstriyel ar-ge projesi çıktısı olan bir yazılım ürün hattı mimarisindeki kullanımı gösterilmiştir. İlgili çalışmalar kısmında, literatürdeki benzerleri ile çalışma kıyaslanmıştır. Son bölüm ise önerilen yöntemin bir değerlendirmesini ve ileriye yönelik çalışmaları içermektedir.

2 Yetenek Modelleme ve Alana Özgü Diller

Clafer (*class, feature, reference*), yazılım ürün hatlarının modellenmesi ve analizi için geliştirilmiş metinsel bir dildir. Dil, yetenek modelini, yazılım ürün hattı mimarisini, ürün hattı mimari şablonlarını ve ürün konfigürasyonunu bir arada tarifleyebilmektedir. Üstmodel ile yetenek modellemeyi iki dilin anlam bütünlüğünü koruyarak birleştirmektedir [1,2]. Ayrıca kümeleri ve ilişkileri temel alan söz dizim ve semantik sayesinde SAT çözücülerine Alloy [6] aracı üzerinden dönüşüm sağlanarak modellerin otomatik analiz edilmesi sağlanmaktadır. Alloy, küme teorisini ve ilişkisel mantığı temel alan formal bir yapısal modelleme dilidir. Birinci Derece Yüklem Mantığı'ndan (İng. *First-Order Predicate Logic*) yararlanarak yüklemleri (İng. *predicate*) büyük *Boolean* ifadelere çevirebilmektedir. Daha sonra da bu ifadeler SAT çözücüleri tarafından otomatik olarak analiz edilebilmektedir. Dolayısıyla mantıksal formüller araca girdi olarak verildiğinde, Alloy Analyzer, altındaki KodKod [14] ilişkisel model bulucusu yardımıyla bu formülü karşılayacak modelleri bulmaya çalışmaktadır.

Bu bildiride önerilen yöntem, SAT çözücüsü desteği olan ve bir metinsel yetenek modelleme dili olan Metin Tabanlı Değişkenlik Dili ile de (İng. *Text-based Variability Language*) (TVL) [8] gerçekleştirilebilirdi. Clafer'in tercih edilmesinde, dilin

TVL'e göre sağladığı çoklu örnekleme, ilişkisel referans kurma ve kalıtım desteği rol oynamıştır [9]. Bu yönden iki dilin sözdizimi incelendiğinde TVL'den Clafer'e dönüşüm olanaklı iken tersi kısmi olarak mümkündür. Böylelikle, Clafer'in TVL dilini kapsadığı çıkartılabilir.

Yöntem bünyesinde kullanılabilecek ve EMF ile entegre dilebilecek bir başka yetenek modelleme dili ise Nesne Yönetim Grubu'nun (İng. *Object Management Group)* (OMG)[1] standartlaştırma sürecinde olan Genel Değişkenlik Dili'dir (İng. *Common Variability Language*) [10]. Bu bildiri yazıldığı tarihte henüz başlangıç sunuşu yapılmış ve araç desteği sağlanmadığı için önerilen yöntem bünyesinde değerlendirilmemiştir. Aynı zamanda OMG'nin önerisinde bir model analiz yöntemi ihtiyacı belirtilmemiştir.

Yapılan endüstriyel çalışmalar ve akademik araştırmalar, DSL'lerin yazılım ürün hatlarındaki yetenek modelleme dilleri ile programlama dilleri arasındaki boşluğu kapatmada oldukça önemli olduğunu göstermektedir [3,4]. DSL'ler, yetenek modelleme dilleri ile belirtmesi zor uygulama mantığını ifade etmede problem uzayı seviyesinde kullanılabilir [4]. Aynı zamanda bir DSL bünyesinde geliştirilen program ya da modellerin değişkenliğini yönetmekte yetenek modelleri ile birlikte kullanılabilirler.

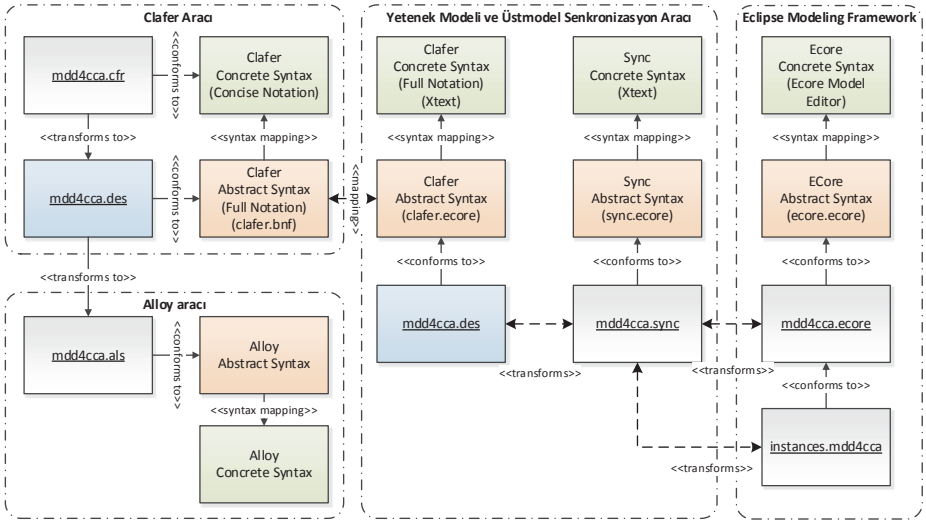
Bir DSL oluşturmak için dilin somut ve soyut sözdizimlerinin tanımlanması gerekmektedir. Yaygın olarak başvurulan soyut sözdizimi oluşturma yöntemleri arasında gramer ya da üstmodel geliştirme gelmektedir. Açık kaynak kodlu Eclipse Modelleme Platformu (İng. *Eclipse Modeling Platform*) bünyesinde sunduğu somut ve soyut sözdizimi geliştirme araçları ile bu iki tekniğin kullanımını desteklemektedir [18]. Eclipse EMF teknolojisi, ECore ile üstmodelleme sağlar [7]. Bir başka Eclipse projesi olan Xtext[2] çatısı ise programlama dilleri ve alana özgü dillerin geliştirilebilmesi için EBNF gramer oluşturma tekniği sunmaktadır. Araç, ECore ve EBNF arasında dönüşüm yapabilmektedir. Bu sebeplerden ötürü oluşturduğumuz yöntemler için Eclipse Modelleme Projesinin kullanımı tercih edilmiştir.

3 Yöntem ve Mimari

Yöntemin başarısı genel olarak Clafer ve Ecore dilleri arasında yatay dönüşüm kurallarının etkin bir şekilde tanımlanmasına bağlıdır. Aynı zamanda bir başka önemli konu ise izlenebilirliğin sağlanması ve geliştirilen alana özgü dillerin kolaylıkla Clafer aracı ile analiz edilebilmesidir. Bu sebeple modelden modele bir dönüşüm dili olan ATL [17] gibi bir dil kullanmak yerine iki dil arasında senkronizasyon sağlamak amacıyla Şekil 1'deki mimari geliştirilmiştir. Bir sınıf modelleme gösterim sistemi olan Ecore, bir OMG standardı olan Üst Nesne Binası'nın (İng. *Meta-Object Facility*) (MOF) modelleme dillerinin üstmodellerini ifade etmek için geliştirilen ve bir alt kümesi olan *Essential* MOF'un eşleniğidir. ECore üstmodeli kendi kendini tanımla-

[1] Object Management Group: http://www.omg.org/
[2] Xtext framework, http://eclipse.org/Xtext

yabilen bir yapıdadır. Dönüşümlerimiz için hem hedef hem de kaynak üstmodelidir ve Eclipse EMF projesi içerisinden elde edilmiştir (ecore.ecore[3]).



Şekil. 1. Yöntem Mimarisi

Clafer aracı yetenek modellerini ve konfigürasyonlarını otomatik olarak Alloy diline çevirebilmektedir. Bu sayede geliştirilen bir üstmodel ve model örnekleri otomatik olarak Clafer diline dönüştürülerek sınanabilmektedir. Bu sebeple geliştirdiğimiz Eclipse eklentisi, en güncel Alloy ve Clafer araçlarını bünyesinde tekrar dağıtmaktadır. Kullanıcı bu araçların entegrasyonu ile uğraşmamaktadır. Clafer aracı iki gösterim kullanmaktadır; ilki kısa ve öz sözdizimi içeren *concise notation*'dır ve *.cfr* uzantılı dosyalarda saklanmaktadır (Tablo 1.a). Diğeri ise *.des* uzantılı dosyalarda saklanan ve tam notasyonu temsil eden *desugared notation*'dır. (Tablo 1.b). Bu notasyon bir BNF gramer ile gösterilebilmektedir. Bu durumda, "*concise notation*"'ın somut sözdizimini "*desugared notation*"'ın ise soyut söz dizimini oluşturduğu çıkartılabilir. Clafer aracı, kendi yorumlayıcısına "cfr" uzantılı bir modeli iletmeden önce onu "des" uzantılı bir modele ön-işlemden geçirerek vermektedir. Kısıt dili içermeyen soyut sözdizimi Şekil 2'de görülmektedir. Clafer yorumlayıcı ise sözdizimine özgü kontrolleri ve isim çözümleme işleminden geçirilmiş modelleri Alloy diline dönüştürerek küme teorisine ve ilişkisel mantığa özgü olarak anlamlandırır. Clafer somut sözdizimi ve kısıt dili, çalışmamızda [2]'den elde ettiğimiz bir BNF grameri (clafer.bnf[4]) ile temsil edilmektedir. Bu gramer, EMF entegrasyonu için bir ECore üstmodeline dönüştürülmüş (clafer.ecore[5]) ve Clafer'in bir IDE sunmayışından ötürü Xtext aracı yardımı ile Eclipse uyumlu bir metinsel somut sözdizimi oluşturulmuştur.

---

[3] Tam ECore üstmodeli: http://www.mdd4cca.com/metamodels/ecore.ecore

[4] Tam BNF: http://www.mdd4cca.com/grammars/clafer.ebnf

[5] Tam Clafer üstmodeli: http://www.mdd4cca.com/metamodels/clafer.ecore

⟨*Clafer*⟩ ⇒ ⟨*Abs*⟩ ⟨*GCard*⟩ string ⟨*Super*⟩ ⟨*Target*⟩ ⟨*Card*⟩ ⟨*Elements*⟩
⟨*Abs*⟩ ⇒ | **abstract**
⟨*Elements*⟩ ⇒ { ⟨*EIList*⟩ } ⟨*EIList*⟩ ⇒ | ⟨*Element*⟩ ⟨*EIList*⟩
⟨*Element*⟩ ⇒ ⟨*Clafer*⟩ | ⟨*Constraint*⟩
⟨*Super*⟩ ⇒ | : string
⟨*Target*⟩ ⇒ | ⟨*Kind*⟩ string
⟨*Kind*⟩ ⇒ → | ↠
⟨*GCard*⟩ ⇒ | **xor** | **or** | **mux** | **opt** | ⟨*NCard*⟩
⟨*Card*⟩ ⇒ | ? | + | ∗ | ⟨*NCard*⟩
⟨*NCard*⟩ ⇒ integer .. ⟨*ExInteger*⟩
⟨*ExInteger*⟩ ⇒ ∗ | integer

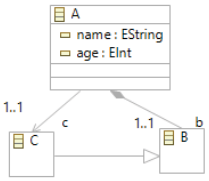
Şekil. 2. Clafer yetenek modelleme dili soyut sözdizimi (kısıt dili içermeyen) [1] (clafer.bnf[4])

Tablo 1. Clafer, Alloy ve Ecore dönüşümü için bir örnek senaryo gösterimi

a. Case1.cfr (Clafer Concise Notation)

```
abstract A
    name : string
    age : integer
    b : B
    c -> C
abstract B
abstract C : B
```

b. Case1.ecore (ECore Notation)



c. Case1.des (Clafer Desugared Notation)

```
abstract 0..* c1_A : clafer 0..* {
   0..* c2_name : string 1..1 { }
   0..* c3_age : integer 1..1 { }
   0..* c4_b : c15_B 1..1 { }
   0..* c5_c ->> c16_C 1..1 { }
   [all disj x; y:
        this.c5_c | x.ref != y.ref]
}
abstract 0..* c15_B : clafer 0..* { }
abstract 0..* c16_C : c15_B 0..* { }
```

d. Case1.des (Alloy Notation)

```
open util/integer
pred show {}
run  show for 1 but 1 c15_B
abstract sig c1_A
{ r_c2_name : one c2_name
, r_c4_b : one c4_b
, r_c5_c : one c5_c }
{ all disj x, y : this.@r_c5_c |
(x.@ref) != (y.@ref) }
sig c2_name { ref : one Int }
{ one @r_c2_name.this }
sig c3_age { ref : one Int }
{ one @r_c3_age.this }
sig c4_b extends c15_B {}
{ one @r_c4_b.this }
sig c5_c { ref : one c16_C }
{ one @r_c5_c.this }
abstract sig c15_B {}
abstract sig c16_C extends c15_B {}
```

Yetenek modelleri ve konfigürasyonları ile üstmodel ve modelleri arasında iki yönlü dönüşümü otomatik olarak gerçekleştirebilmek adına bir dizi dönüşüm kuralının aracın bünyesinde gerçekleştirilmiş olması gereklidir. İki notasyonun soyut sözdizimleri kıyaslandığında EMF çatısı bir modelleme alt yapısı oluşturabilmek adına Clafer'in yetenek modelleme diline göre daha gelişmiştir. Öte yandan Clafer ise Alloy dilinden esinlendiği Birinci Dereceden Yüklem Mantığı kısıt dili sebebiyle oldukça karmaşıktır. Bu farklılıklardan ötürü, tanımlanan dönüşüm kurallarında farklı yöntem-

ler kullanılması ihtiyacı doğmuştur. Clafer ve ECore dilleri arası yatay ve çift yönlü dönüşümler, Xtend[6] dili kullanılarak gerçekleştirilmiştir.

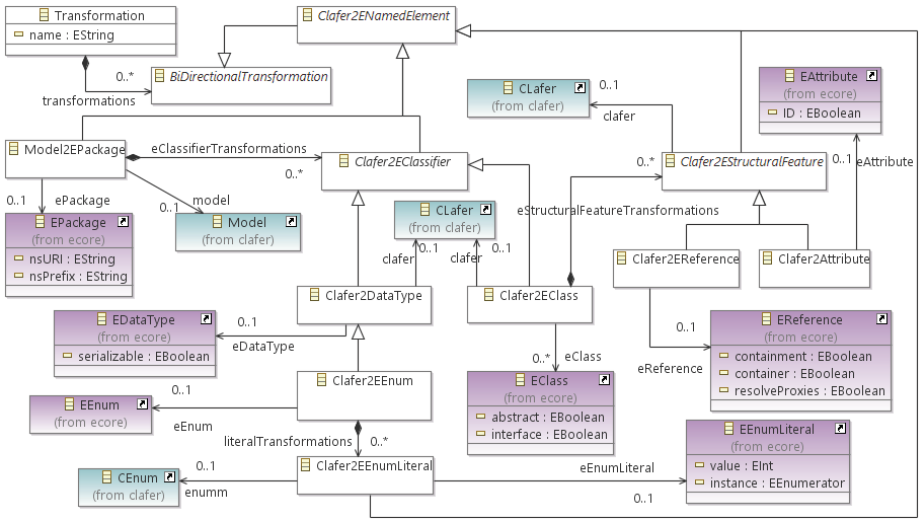
Tablo 2. Clafer ve Ecore genel eşleştirme tablosu (çift yönlü dönüşüm kuralları)

a. Clafer Kavramları	**b. ECore Kavramları**
Clafer	EClass, EEnum, EDataType (Clafer hiyerarşi yapısına göre değişkenlik göstermektedir)
String, Integer (built-in types)	EAttribute (EString, EInt)
Clafer Typing ('abstract EType' şeklinde Clafer üretilir)	EAttribute (EBoolean, EDate, EDouble)
Reference Clafer	EReference
Reference Clafer (':' ve '->')	EReference.IsContainment: EBoolean
Reference Clafer'lar çift yönlü varsayılır. (sync.ecore dönüşümü hatırlar).	EReference.IsOpposite: EBoolean (sync.ecore dönüşümü hatırlar).
Clafer Inheritance	Single Inheritance
Clafer Merging	Multiple Inheritance
Clafer Nesting (sync.ecore dönüşümü hatırlar).	Ecore, iç içe EClass'lar yuvalanmaz (inner EClass) (sync.ecore dönüşümü hatırlar)
Cardinality (lone, some, any, one, interval)	(EAttribute | EReference) .Multiplicity.(LowerBound and UpperBound)
Group Cardinality (xor, or, mux, opt)	*Çeşitli OCL ifadeleri üretilir.*
Karşılık gelecek açık bir anlamı yoktur. (sync.ecore dönüşümü hatırlar)	EClass.IsAbstract: EBoolean (sync.ecore dönüşümü hatırlar).
Singleton Concrete Clafers	ECore Instances

İki dile ait kavramların eşleştirildiği Tablo 2 incelendiğinde bazı kavramların diğerinde tam karşılığının olmadığı ya da bir anlam ifade etmediği görülmektedir (eğik yazılmış olanlar). Dönüşüm kuralları bu gibi bölgelerde önerdiğimiz çeşitli yöntemleri kullanmaktadır (yeni Clafer türetme, Clafer birleştirme, OCL kuralları üretme gibi). Temel dönüşüm kurallarının uygulanışı Tablo 1 ve Tablo 2 karşılaştırılarak incelenebilir. Dördüncü bölümde sunulan örnek senaryoda bu tablodaki dönüşümlerin büyük bir kısmı uygulanmıştır. Örnek bir yetenek modeli, Tablo 1.a'da, otomatik olarak aracın ürettiği üstmodel ise Tablo 1.b'de gösterilmiştir. Öte yandan Tablo 1.c Clafer yorumlayıcısına gönderilen notasyonu gösterirken Tablo 1.d ise bu gösterimden üretilen Alloy modelidir. Clafer ve ECore dilleri kavramları arasında tam bir eşleme olmayışı, bazı durumlarda dili oluşturan yapıların sıralanış desenlerine özgü olarak eşleme algoritmaları kurgulanmasına sebep olmuştur. Dönüşümlerdeki gidiş-gelişlerde (İng. *roundtrip)* bilgi kayıpları yaşanmakta, iki iterasyon sonrası ilk modele ulaşılamamaktadır. Bu sebeple bu iki dilin dönüşümlerine özgü olarak bir dil geliştirilmiştir. *sync.ecore* üstmodeli bu dilin soyut sözdizimini oluşturmaktadır. Bu üstmodel, Xtext aracı yardımıyla EBNF gramerine dönüştürülmüş ve Eclipse uyumlu metinsel somut

[6] Xtend Java dialect, http://www.eclipse.org/Xtend/

sözdizimi oluşturulmuştur. Şekil 3'te basitleştirilmiş *sync.ecore* üstmodeli gösterilmektedir. Kimi model elemanları *ecore.ecore* ve *clafer.ecore* üstmodellerinden çağrılmaktadır.



Şekil. 3. Basitleştirilmiş senkronizasyon ve izlenebilirlik üstmodeli (sync.ecore)[7]

4 Yöntemin Kullanıldığı bir Yazılım Ürün Hattı Mimarisi

Geliştirilen yöntem, Bütünleşik İçerik Uygulamalarının Model Güdümlü Geliştirilmesi[8] (mdd4cca) isimli endüstriyel ar-ge projesinin tüm alana özgü görsel ve metinsel dillerinin geliştirilmesinde kullanılmıştır. Projede, endüstride oldukça yaygın olarak kullanılan bir Kurumsal İçerik Yönetim platformu olan *Microsoft SharePoint Foundation*[9] (MSF) üzerinde bir Bütünleşik İçerik Uygulaması[10] (CCA) ya da bir yazılım ürün hattı oluşturabilecek CCA şablonu geliştirilmesi için farklı bakış açıları ile modellemenin sağlanabileceği görsel bir araç setinin hazırlanması faaliyetleri yürütülmüştür. Alana özgü kavramlar ve aralarındaki ilişkileri görsel ve metinsel modelleme araçlarına mümkün olduğunca yansıtmak için önce alan kavramlarına dair bir kavram sözlüğü çalışması yapılmış, endüstriyel senaryolarda sıklıkla karşılaşılan ihtiyaçlar değerlendirilmiş, gerekli pazar analizi çalışmaları yürütülmüştür. Sonrasında ise Clafer yetenek modelleri bu kavramlar üzerinden oluşturulmuştur. Geliştirilen yetenek modeli, çeşitli örnek konfigürasyonlar eşliğinde, Clafer aracının sağladığı Alloy diline dönüşüm ile analiz edilmiştir. Bu yetenek modellerinden, önerilen yöntem yardımıyla

---

7 Tam *sync.ecore* üstmodeli: http://www.mdd4cca.com/metamodels/sync.ecore

8 Model-driven Development for Composite Content Applications: http://www.mdd4cca.com.

9 Microsoft SharePoint Foundation: http://technet.microsoft.com/library/cc303422.aspx

10 Composite Content Applications: https://www.gartner.com/doc/1286414/

problem uzayını sınırlandıracak çeşitli üstmodeller türetilmiştir. Projenin geliştirme çevrimlerinde üstmodellerde yapılan değişiklikler otomatik olarak önerilen yöntem yardımıyla yetenek modellerine yansıtılabilmiştir. Öte yandan, farklı alanlardaki çeşitli CCA örneklerine ait proje kodları incelenmiş ve belirlenen mimari üzerinde benzerlik ve değişkenlik analizi yapılmıştır ve değişkenliklere ait tüm bilgiyi modelleyebilecek bir çözüm uzayı üstmodeli üretilmiştir. Clafer dili, yalınlığı sayesinde çözüm uzayı soyut sözdiziminin tasarımında da kullanılmıştır. Önerilen yöntem sayesinde Clafer modellerinden ECore üstmodeline geçiş otomatik olarak sağlanmıştır. Bilindiği üzere problem uzayı alan uzmanlarına yakın iken, çözüm uzayı ise mimariye yönelik üretilecek bileşenler içermesi sebebiyle geliştiricilere daha yakındır [3]. İki üstmodel arasındaki soyutlama boşluğu ise problem uzayı üstmodelinden çözüm uzayı üstmodeline model dönüşüm kuralları ile otomatikleştirilerek kapatılmıştır. Çözüm uzayı modellerinden ise platforma ait çeşitli yazılım bileşenleri otomatik üretilebilmektedir.

Tablo 3. Yazılım ürün hattının içerik bakış açısına ait yetenekleri (mdd4cca.cfr)[11]

a. Clafer Yetenek Modeli

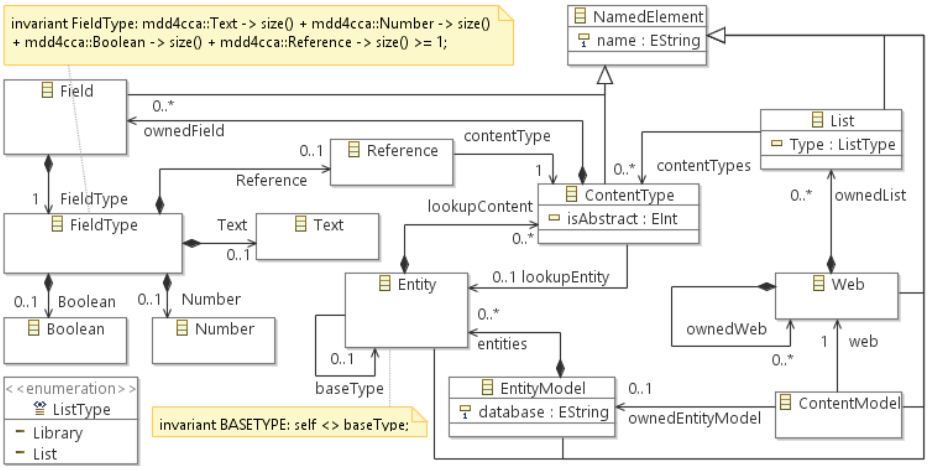
```
abstract NamedElement
    name: string

abstract ContentModel : NamedElement
    web -> Web +
    ownedEntityModel->EntityModel?

abstract Web : NamedElement
    ownedList : List *
    ownedWeb : Web *

abstract List : NamedElement
    contentTypes -> ContentType *
    xor Type
         Library
         List

abstract ContentType : NamedElement
    isAbstract : integer ?
    ownedField : Field *
    lookupEntity -> Entity ?

abstract Field: NamedElement
    xor Type
        Number
        Boolean
        Text
        Reference
            contentType -> ContentType

abstract EntityModel : NamedElement
    database: string
    entities: Entity *

abstract Entity: NamedElement
    baseType -> Entity ?
    lookupContent -> ContentType *
```

b. Clafer Konfigürasyon

```
EgeLibrary : ContentModel
    [name = "EGE_Kutuphanesi"]
    [web = LibraryWebSite]

LibraryWebSite : Web
    [name = "Kutuphane_Web_Sitesi"]
    TezKitapligi : List
        [Library][name="Tez_Kitapligi"]
        [contentTypes
          = Dissertation, MasterThesis]
    OgrenciListesi : List
        [List][name="Ogrenci_Listesi"]
        [contentTypes = Student]

Student : ContentType
    [name = "Ogrenci"]
    studentNo : Field
        [Number][name = "Ogrenci_No"]

abstract Thesis: ContentType
    [name = "Tez"]
    thesisName: Field
        [Text][name = "Tez_Adi"]
    student: Field
        [Reference][name = "Ogrenci"]
        [contentType = Student]

MasterThesis: Thesis
Dissertation: Thesis
```

[11] Tam yetenek modeli: http://www.mdd4cca.com/features/mdd4cca.cfr

İçerik modelleme dili, SharePoint platformu üzerinde koşacak bir projenin içeriğinin tanımlandığı modeldir. Bu dilin Clafer tabanlı yetenek modelinin (yer limitleri sebebiyle) basitleştirilmiş bir hali Tablo 3.a'da sunulmuştur. Bu model, bütünleşik içerik uygulamaları için bir alan modeli sunarken bünyesinde analiz amaçlı bir ya da birden çok alana uygun konfigürasyon ile birlikte tasarlanmıştır. Örnek bir konfigürasyon Tablo 3.b'de gösterilmiştir. Bu basitleştirilmiş örnekte, SharePoint İçerik modelleme ile EDM[12] formalizmini bütünleştiren bir dile ait yetenekler gösterilmiştir. Alan uzmanının, içerik modeli bünyesinde bir *Web* varlığı tanımlaması zorunludur. *Web* içerisinde istediği kadar *List* varlığı ekleyebilir. Her bir liste içerisinde istediği kadar içerik türü (*content type)* tanımlayabilmektedir.



Şekil. 4. Otomatik üretilen mdd4cca üstmodeli (mdd4cca.ecore)[13]

Dönüşüm iz bilgileri araç tarafından, senkronizasyon modelinde otomatik olarak saklanmakta ve geliştirilen metinsel somut sözdizimi yardımıyla dönüşüm kurallarında değişkenlik uygulanabilmektedir. Ortaya çıkan örnek model, *sync.ecore* üstmodeline uygundur. Araç, *mdd4caa.cfr* yetenek modelinden, Şekil 4'da yer limitleri sebebiyle basitleştirilmiş hali gösterilen *mdd4cca.ecore* üstmodelini üretmiştir. Konfigürasyona dair Clafer'leri ise *mdd4cca.ecore* üstmodeline uygun örnek modeller olarak algılamış ve *instances.mdd4cca*[14] dosyası içerisinde XMI serileştirmiştir.

5 İlgili Çalışmalar

[1,2] çalışmalarında Clafer yapıları UML sınıf diyagram yapıları ile karşılaştırılmıştır. Ancak belirtilen dönüşüm kuralları sadece Clafer içerisinde alan modelleme ya da

[12] Entity Data Model: http://msdn.microsoft.com/library/ee382825.aspx
[13] Tam mdd4cca üstmodeli: http://www.mdd4cca.com/metamodels/mdd4cca.ecore
[14] Tam örnek modeller: http://www.mdd4cca.com/examples/instances.mdd4cca

bileşen modeli tanımlama yeteneğinin gösterilmesine yöneliktir. Sunduğumuz yöntemin amacına uygun olarak EMF uyumlu bütün bir süreç ve araç sunmamaktadır. Dönüşüm kuralları sadece statik yapısallığı kapsamaktadır. Ayrıca ECore örnek modelleri ve ileri yönlü bir dönüşüm sonrası ECore modelinde oluşması gereken OCL kısıtları bu çalışmalarda incelenmemiştir. Clafer ile UML yaklaşımı karşılaştırdığında Clafer sözdiziminin UML sınıf diyagram dilinin soyut sözdizimi olan MOF üstmodelini kapsadığı düşünülebilir ancak MOF üstmodeli daha önceki bölümde gösterildiği gibi yapısal modelleme konusunda daha fazla bilgi içermektedir. ECore üstmodelinin MOF'un bir gerçekleştirimi olduğunu düşünürsek, belirttiğimiz sebepten ötürü bir ECore modelinden Clafer modeline geçişte bilgi kayıpları yaşanacaktır. Bu açıdan da bu çalışmalar bir çözüm önermemektedir.

Clafer'i geliştiren Kanada, Waterloo Üniversitesi *Generative Software Development Lab,* aynı zamanda ECore üstmodellerini Clafer'e dönüştüren *ecore2clafer*[15] isminde bir araç sunmaktadır. Araç tersi dönüşümü desteklememektedir. Çalışma sadece ECore üstmodellerini *Abstract Clafer* yeteneklerine dönüştürebilmektedir. Dönüşüm kuralları kısıtları desteklememektedir. Oysa bir Clafer modelinde tanımlanabilen *Group Cardinalities, Sub-Clafers, Set Partitions* gibi dile özgü kavramlar ECore dilinde olmadığı için bir üstmodelde ancak OCL kısıtları eşliğinde anlamlandırılabilirler.

Önerdiğimiz yönteme amaç yönünden en yakın çalışma, aynı ekibin Clafer önerilerinden önce geliştirdikleri *Ecore.fmp* aracıdır [12]. Yaklaşım, ECore modellerini yetenek modelleri gibi ele alıp, düzenleme ve örnekleme olanağı sağlamaktadır. Çalışma eleman sayısı temelli yetenek modelleme, (İng. "*Cardinality-based feature modeling*") [11] yaklaşımına uyumlu geliştirilen araca özel bir üstmodel önermektedir. Clafer, bu dili kapsamakla beraber daha anlaşılır, öz ve tam bir semantik kurmaktadır [1,2]. Araç, ECore modellerinde çeşitli *annotation'*lar kullanarak ECore üstmodelini genişletmesiyle model elemanlarına yetenek modeli elemanı gibi davranmaktadır. Bu sebeple bu yöntem sadece araç ile uyumlu bir ECore üstmodelini yetenek modellerine dönüştürebilir. Bu yöntem, senkronizasyon problemlerini çözüyormuş gibi gözükse de iki dilin formalizmini ve anlamını eşitlemektedir. Araç modelleme ve konfigürasyon için kısıtları desteklememekte olup bir çözücü ile entegrasyonu yoktur.

FeatureMapper [15] aracı, yazılım ürün hatlarında yetenek modelleri ile çözüm uzayı modellerini eşleştirerek model güdümlü yazılım mühendisliği süreci bünyesinde yetenek modellerinin kullanımını hedefleyen bir çalışmadır. Yetenek modellerindeki yetenekler ile onların ECore modellerindeki gerçekleştirimlerine eşleştirebilmek için bir üstmodel önermektedir. Bu üstmodel yardımı ile bir yeteneğin seçilip seçilmemesine göre, eşleştirilen çözüm uzayı modelinde eksiltme yapmaktadır. *Ecore.fmp* çalışmasındaki gibi eleman sayısı temelli yetenek modellerini destekleyen araca özel bir üstmodel kullanmaktadır. Bu sebeple *Ecore.fmp* ile benzer zayıflıkları paylaşmaktadır. Aynı zamanda önerdiğimiz yöntem, problem uzayı ya da çözüm uzayında geliştirilen EMF tabanlı alana özgü dillerin, Clafer diline eşleştirerek geliştirilmesi (veya tam tersinin) sağlanabilmesi yönünden farklılık göstermektedir.

---

[15] Ecore to Clafer Translator: http://gsd.uwaterloo.ca/ecore2clafer

[16] çalışmasında emniyet kritik bir aviyonik yazılımın doğal dilde yazılmış gereksinimleri mantıksal ifadelere çevrilmiş, problem alanı Alloy dili ile modellenmiş, tutarsızlıklar ve eksiklikler bakımından analiz edilmiştir. Sunulan yöntem bünyesinde karşılaşılan zorluğun Alloy dili ile modellemede harcanan efor olduğu belirtilmiştir. Bu bildiride önerilen yöntem, bu yolla analiz edilebilecek dillerin Alloy dili yerine gereksinim modellemeye daha yakın olan Clafer ile yapılmasını olanaklı kılmaktadır.

6 Sonuç ve İleriye Yönelik Çalışmalar

Bu bildiride, metinsel bir yetenek modelleme dili olan Clafer ile bir üstmodelleme dili olan ECore arasında senkronizasyon sağlayan bir yöntem önerilmiştir. Yöntem yetenek modelleri ve konfigürasyonları ile üstmodel ve modelleri arasında iki yönlü dönüşümü otomatik olarak gerçekleştirebilmektedir. Bu iki gösterim sistemi arasındaki dönüşüm kurallarına özgü olarak geliştirilen dil yardımıyla bir önceki dönüşüm kuralları hatırlanabilmektedir. Böylelikle Clafer ve ECore arasındaki anlamsal farklılıktan ötürü dönüşümler sonucu ortaya çıkan bilgi kaybı giderilmiştir. Aynı zamanda dönüşüm kurallarındaki değişkenlik aynı model yardımıyla yönetilebilmekte, geliştiriciye dönüşüm kurallarında seçim yapabilme olanağı sağlamaktadır. Geliştirilen araç seti, endüstriyel senaryolarda uygulanmak üzere Eclipse eklentisi olarak dağıtılmaktadır[16].

Çalışmamalarımız Clafer dilinin DSL geliştiriminde önemli iki katkısının da araştırılmasını ve geçerlemesini sağlamıştır. Birinci katkı ECore ile üstmodelleme yapmak yerine ifade gücü yüksek ve basit Clafer dilini kullanarak bir DSL'in soyut sözdizimin tasarlanmasının mümkün olduğunun gösterilmesidir. Yaptığımız dönüşümler, Clafer'in ECore için yeni bir metinsel somut sözdizimi olarak kullanılabileceğini de göstermektedir. İkinci önemli katkı ise Clafer dilinin SAT ve CSP çözücüleri entegrasyonu sayesinde, geliştirilen bir DSL'i ayrıca bir çözücü ile eşlemeye gerek kalmadan doğrulama ve kontrol etme olanağı sağlayabilmesinin gösterilmesidir.

Önerdiğimiz yöntem Clafer yetenek modellerinden gelen belli bir takım kısıtları ECore OCL ifadelerine dönüştürebilmektedir. Bu dönüşüm kurallarının beklenen düzeyde olmaması Clafer kısıt dilinin Alloy tabanlı olmasından ötürüdür. Alloy ile UML ve OCL arasındaki çift yönlü dönüşüm probleminin [13,19] çözülmesi ve çözümün endüstrileştirilmesi çalışmalarımız devam etmektedir.

Teşekkür. Bu çalışma, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) Teknoloji ve Yenilik Destek Programları Başkanlığı (TEYDEB) 1501 Sanayi Ar-Ge Projeleri Destekleme Programı'nca desteklenen 3110712 numaralı "Bütünleşik İçerik Uygulamalarının Model Güdümlü Geliştirilmesi" projesi kapsamında yürütülmüştür.

[16] Araç Eclipse güncelleme sitesi: http://www.mdd4cca.com/updates/clafer2ecore

7 Kaynakça

1. Bąk, K., Czarnecki, K., Wąsowski, A.: Feature and meta-models in Clafer: mixed, specialized, and coupled. In: Software Language Engineering, pp.102-122. Springer Berlin Heidelberg (2011)
2. Bąk, K: Modeling and Analysis of Software Product Line Variability in Clafer. Diss. University of Waterloo (2013)
3. Groher, I., Voelter, M.: Aspect-oriented model-driven software product line engineering. In: Transactions on aspect-oriented software development. Springer Berlin (2009)
4. Voelter, M., Visser, E.: Product line engineering using domain-specific languages. In: 15th International Software Product Line Conference (SPLC) pp. 70-79. IEEE (2011)
5. Tekinerdogan, B.: First Turkish software product line engineering workshop summary. ACM SIGSOFT Software Engineering Notes 37, no. 6, pp. 30-34 (2012)
6. Jackson, D.: Alloy: a lightweight object modelling notation. ACM Transactions on Software Engineering and Methodology (TOSEM) 11, no. 2, pp 256-290 (2002)
7. Steinberg, D., Budinsky, F., Merks, E., & Paternostro, M.: EMF: eclipse modeling framework. Pearson Education (2008)
8. Classen, A., Boucher, Q., Heymans, P.: A text-based approach to feature modelling: Syntax and semantics of TVL. Science of Computer Programming. pp 1130-1143. (2011)
9. Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., Wąsowski, A.: Cool features and tough decisions: a comparison of variability modeling approaches. In: 6th International workshop on variability modeling of software-intensive systems. pp.173-182. ACM (2012)
10. Haugen, O., Moller-Pedersen, B., Oldevik, J., Olsen, G. K., Svendsen, A.: Adding standardized variability to domain specific languages. In:12th International Software Product Line Conference, 2008, pp. 139-148. IEEE (2008)
11. Czarnecki, K., Helsen, S., Eisenecker, U.: Formalizing cardinality-based feature models and their specialization. Software process: Improvement and practice 10(1) pp 7-29 (2005)
12. Stephan, M., Antkiewicz, M.: Ecore.fmp: A tool for editing and instantiating class models as feature models. University of Waterloo, Technical Report 8, (2008)
13. Kuhlmann, M., Gogolla, M.: From UML and OCL to relational logic and back. In: Model Driven Engineering Languages and Systems. pp. 415-431. Springer Heidelberg (2012)
14. Torlak, E., Jackson, D.: Kodkod: A relational model finder. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 632-647. Springer Berlin Heidelberg (2007)
15. Heidenreich, F., Kopcsek, J., Wende, C.: FeatureMapper: mapping features to models. In: 30th international conference on Software engineering, pp. 943-944. ACM (2008)
16. Ata, B., Oğuztüzün, H., Yüceer, R. E.: Alloy Analyzer Kullanarak Emniyet Kritik Aviyonik Yazılım Gereksinimlerinin İncelenmesi. In: 4. Ulusal Yazılım Mühendisliği Sempozyumu, pp. 51-55, İstanbul, Turkey (2009)
17. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming, 72(1-2): pp. 31-39 (2008)
18. Gronback, R. C.: Eclipse modeling project: a domain-specific language (DSL) toolkit. Pearson Education (2009)
19. Anastasakis, K., Bordbar, B., Georg, G., & Ray, I.: UML2Alloy: A challenging model transformation. In: Model Driven Engineering Languages and Systems, pp. 436-450. Springer Berlin Heidelberg (2007)
20. Beuche, D.: Modeling and building software product lines with pure::variants. In: 16th International Software Product Line Conference-Volume 2, pp. 255-255. ACM (2012)
21. Krueger, C. W.: The biglever software gears unified software product line engineering framework. In: Software Product Line Conference (SPLC 12), pp. 353-353. IEEE (2008)