Yazılım Mühendisliğinde Ontolojilerin Kullanımı

Görkem Giray[1] Murat Osman Ünalır[2]

[1,2]Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir
[1]e-posta: gorkemgiray@mail.ege.edu.tr [2] e-posta: murat.osman.unalir@ege.edu.tr

Özetçe

Bilim ve teknolojideki gelişmeler yazılımı, hayatımızın ayrılmaz bir parçası haline getirmiştir ve kullanıcıların yazılımdan beklentilerini arttırmıştır. Bu durum, yazılımın ve yazılım geliştirme süreçlerinin daha da karmaşıklaşmasına yol açmıştır. Bu karmaşıklığı yönetebilmek için, modellerin yazılım geliştirme süreçlerinin ana yapısı olarak kullanılması benimsenmektedir. Diğer taraftan, anlamsal web çalışma alanındaki gelişmeler, bilgisayar bilimleri için yeni olmayan ontolojilerin, farklı alanlarda kullanımını tetiklemiştir. Ontolojilerin kullanılabileceği alanlardan birisi de yazılım mühendisliğidir. Özellikle, bir konu alanını biçimsel olarak temsil etmesi ve yeniden kullanılabilirliği desteklemesi nedeniyle, ontolojilerin gereksinim mühendisliğinde kullanımı önerilmektedir. Modelleri ana yapılar olarak kabul eden yazılım geliştirme yaklaşımlarında, gereksinimlerin biçimsel olarak temsil edilebilmesi ve gereksinimler ile daha alt seviyedeki analiz ve tasarım modelleri arasındaki izlenebilirliğin sağlanması önemlidir. Bunun yanında, mevcut ontolojilerin değiştirilerek ya da genişletilerek kullanılması, yazılım geliştirme etkinliklerinin maliyetlerini düşürmek için kullanılabilecek yöntemlerden birisidir. Bu amaca ulaşabilmek için yazılım mühendisliğindeki ve ontolojilerin temelindeki farklılıkları irdelemek gerekmektedir. Bu farklılıklar, katmanlı mimari, açık-kapalı dünya yaklaşımları ve birlikte işlerliğe yaklaşım başlıkları altında toplanabilir.

1. Giriş

Yazılım mühendisliğinin amaçlarından birisi, yazılım geliştirmedeki karmaşıklığı azaltmaktır. Bunun için önerilen yöntemlerden birisi soyutlama seviyesinin arttırılmasıdır. Soyutlama seviyesinin arttırılması, yazılım geliştirme sürecinde üretilen yapıların mümkün olduğu kadar problem alanına yakınlaştırılmasıyla mümkündür. Bunun için yazılım geliştirme süreçlerinde modellerin ana yapılar olarak kullanılması gelecek nesil yazılım geliştirme metodolojilerinin temelini oluşturmaktadır. Modelleri, yazılım geliştirme etkinliklerinin odağına yerleştiren yaklaşıma "Model Güdümlü Mühendislik" veya "Model Güdümlü Yazılım Geliştirme" isimleri verilmektedir. Object Management Group (OMG) bu kapsamdaki çalışmalarını "Model Güdümlü Mimari", Microsoft bu kapsamdaki çalışmalarını ise "Yazılım Fabrikaları" adı altında toplamaktadır.

Anlamsal web çalışma alanı ise İnternet'in hayatımızdaki öneminin ve elektronik ortamdaki veri hacminin önlenemez artışıyla birlikte oldukça ilgi çekmeye başlamıştır. Bu çalışma alanının merkezindeki kavramlardan birisi ontolojilerdir. Ontolojiler, bilginin anlamının biçimsel temsilini sağlayarak, şu anda tamamen insanlar tarafından yapılan elektronik ortamdaki bilginin anlamı üzerinde çıkarsama yapma işini, kurallar ve çıkarsama motorları yardımıyla, bir ölçüde bilgisayarlara devretmeyi hedeflemektedir.

Yazılım mühendisliğindeki ve anlamsal web çalışma alanındaki bu gelişmelerin birbirini tamamlayıcı nitelikte olduğu birçok araştırmacı tarafından irdelenmektedir. Bundan dolayı ontoloji geliştirme etkinliklerinin yazılım mühendisliği ile nasıl bütünleştirilebileceği konusunda çalışmalar yapılmaktadır [1,2].

Bu bildiride, anlamsal web ve yazılım mühendisliği çalışma alanlarının birbirlerine yapabilecekleri katkılar irdelenmiştir. İkinci ve üçüncü bölümlerde, anlamsal web ve yazılım mühendisliği çalışma alanları tanıtılmıştır ve bu çalışma alanlarında yaşanan gelişmeler özetlenmiştir. Yazılım mühendisliğinde modellemenin önemi üzerinde durulmuştur ve model güdümlü mühendislik yaklaşımı açıklanmıştır. Model güdümlü mühendisliğin en çok bilinen iki gerçekleştirimi olan OMG'nin Model Güdümlü Mimarisi ile Microsoft'un Yazılım Fabrikaları kısaca tanıtılmıştır. Dördüncü bölümde ise ontolojilerin yazılım mühendisliğindeki kullanımı için önemli bir örnek oluşturan "Model Güdümlü Mimari Tabanlı Ontoloji Mühendisliği", bu kapsamda OMG tarafından geliştirilen Ontology Definition Metamodel (ODM) ve gereksinim mühendisliğinde ontolojilerin kullanımları üzerinde durulmuştur. Beşinci bölümde, bu bildirinin temel katkılarından birisi olarak, yazılım mühendisliğindeki ve ontoloji mühendisliğindeki temel yaklaşım farklılıkları irdelenmiştir. Altıncı bölümde bildirinin sonuçları özetlenmiştir.

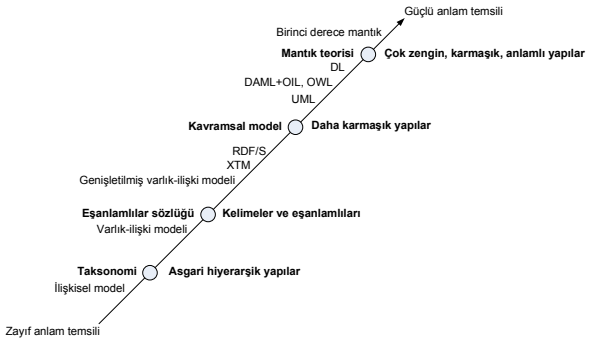
2. Anlamsal Web ve Ontolojiler

Anlamsal web araştırma alanının amacı, bilginin anlamının biçimsel olarak temsil edilerek, bilginin sadece insanlar tarafından değil bilgisayarlar tarafından da anlamsal seviyede işlenebilmesini sağlamaktır.

Bilginin anlamsal temsili için ontolojiler kullanılmaktadır. Ontoloji kavramının en çok kullanılan tanımı "paylaşılan bir kavramsallaştırmanın biçimsel ve net bir belirtimi"dir [3]. Uschold ve Gruninger bu tanımda geçen her kelimeyi açıklayarak bu kısa tanımın ifade ettiği uzun ontoloji tanımını açıklamışlardır [4]. "Kavramsallaştırma", insanların dünyadaki varlıklar üzerine nasıl düşündüklerinin soyut bir modelini ifade eder. Bu soyut model genellikle özel bir konu alanı ile sınırlandırılmıştır. "Net bir belirtim" ise soyut modeldeki kavramlara ve ilişkilere net isimler verildiği ve net tanımlar yapıldığı anlamına gelmektedir. Bir kavramın ya da ilişkinin tanımı, o terimin anlamının ifade edilmesidir. Bir başka deyişle, bir terimin diğer terimlerle ilişkisinin nasıl olacağını belirtir. "Biçimsel" ifadesi, anlam tanımının biçimsel bir dille temsil edildiğini ve böylece tanım üzerindeki belirsizliklerin, farklı anlam çıkarma olasılıklarının ortadan kaldırıldığını ifade etmektedir. Bundan dolayı biçimsel temsil, otomatik çıkarsama yapma imkanını sağlamaktadır. "Paylaşılan" kelimesi ise ontolojilerin, farklı

uygulamalar ve topluluklar arasında yeniden kullanımı amaçladıklarını ve desteklediklerini ifade etmektedir.

Ontolojiler, sahip oldukları temsil gücüne göre sınıflandırılırlar. Şekil 1'de Daconta ve arkadaşlarının oluşturduğu, bilgi modellerini ve modelleme dillerini anlamsal zenginliklerine göre derecelendiren bir ontoloji yelpazesi gösterilmektedir [5].



Şekil 1: Ontoloji yelpazesi.

Şekil 1'de görüldüğü gibi, ilişkisel bir veritabanı tasarımı için hazırlanmış varlık-ilişki modeli, yazılım geliştirme sürecinde Unified Modeling Language (UML) ile oluşturulmuş bir sınıf diyagramı ya da Web Ontology Language (OWL) ile oluşturulmuş kavramsal bir model de ontoloji olarak adlandırılmaktadır. Fakat bu modellerin temsil güçleri birbirinden farklıdır. Ontoloji yelpazesinde sol aşağıdan sağ yukarıya doğru gidildikçe temsil gücü artmaktadır. Böylece kavramlar arasında daha fazla ilişki temsil edilebilmektedir. Fakat temsil gücü arttıkça gereksinim duyulan çıkarsama zamanı da artmaktadır. Temsil gücü belirli bir sınırı aştığında yapılan çıkarsamaların kabul edilebilir süreler içinde bitme olasılığı da azalmaktadır.

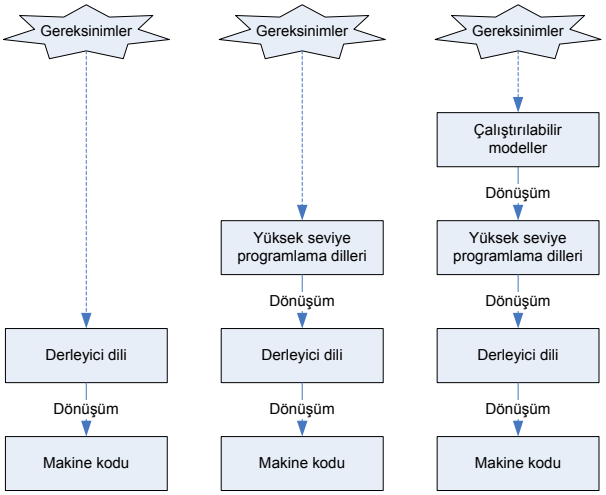
World Wide Web Consortium (W3C), anlamsal web vizyonu çerçevesinde, ontoloji temsil dilleri geliştirmekte ve standartlaştırmaktadır. OWL, W3C'nin ontoloji temsili için geliştirdiği ve standartlaştırdığı dildir. OWL yine W3C tarafından geliştirilmiş olan Resource Description Framework (RDF) ve RDF Schema (RDFS) standartları üzerine temellendirilmiştir.

3. Yazılım Mühendisliğinde Yeni Yaklaşımlar

IEEE Computer Society, yazılım mühendisliğini, sistemli ve ölçülebilir yaklaşımların yazılımın geliştirilmesine, işletilmesine ve bakımına, kısacası mühendislik tekniklerinin yazılıma uygulanması olarak tanımlamaktadır [6]. Yazılım mühendisliğinin en önemli hedeflerinden birisi yazılım geliştirmedeki karmaşıklığın yönetilmesi ve azaltılmasıdır. Karmaşıklığın azaltılması için soyutlama seviyesinin yükseltilmesi en çok önerilen yöntem olarak karşımıza çıkmaktadır. Mellor ve arkadaşları yazılım geliştirmenin tarihçesini, soyutlama seviyesinin yükseltilmesi çabası olarak özetlemektedir [7]. Yazılım geliştirme disiplininde soyutlama seviyesinin yükseltilmesi, yazılım geliştiricinin gerçekleştirim detaylarından mümkün olduğu kadar uzaklaşarak yazılımın problem ve çözüm alanına yaklaşması anlamına gelmektedir.

Yazılım geliştiriciler, derleyici dillerini kullanırken alt seviyede işlerle (bellek yönetimi, vs.) uğraşırken, yüksek seviye programlama dillerinin ortaya çıkmasıyla birlikte bu alt seviye işleri yapma zorunluluğundan kurtulmuşlar ve problem alanına daha çok odaklanma imkanına sahip olmuşlardır. Alt seviyede işler, soyutlama seviyesini yükseltme amacına hizmet eden yazılım geliştirme araçlarının desteğiyle otomatikleştirilerek mümkün olduğu kadar bilgisayara devredilmiştir. Böylece yazılım geliştirme araçları, yazılım geliştirme sürecinde yazılım geliştiriciye daha çok destek olmaya başlamıştır.

Yüksek seviye programlama dilleri, yazılım geliştiricilere, doğal dildeki kelimelere benzer deyimler kullanarak kod üretme olanağını sunar. Yüksek seviye dillerin sunduğu bu deyimlerle üretilen kod, derleyici dili ile yazılan koda göre, gerçek dünyadaki gereksinimlerle daha kolay eşleştirilebilir. Böylece gerçek dünya gereksinimleri ile üretilen kod arasındaki anlamsal boşluk azalmış olur. Soyutlama seviyesinde benzer bir yükselme de çalıştırılabilir modellerin üretilebilmesiyle birlikte sağlanabilmektedir. Şekil 2'de gösterildiği gibi yüksek seviye programlama dillerinin ve çalıştırılabilir modellerin yazılım geliştirmede kullanılmasıyla birlikte üretilen kod ile gerçek dünya gereksinimleri arasındaki anlamsal boşluk giderek azalmaktadır.



Şekil 2: Yazılım geliştirmede otomasyonun artması.

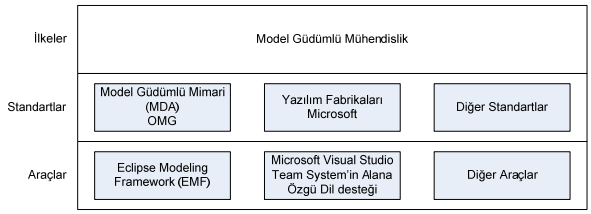
Modeller, bir çok mühendislik probleminin çözümünde çok uzun zamandır kullanılmaktadır. Benzer şekilde modellerin yazılım geliştirme süreçlerinde de kullanılması bazı faydalar sağlamaktadır.

Yazılım mühendisliği bağlamında model, gerçek dünyanın bir bölümünü temsil eder, bir modelleme diliyle ifade edilir ve temsil ettiği gerçek dünyanın bölümü hakkında belirli bir amaç için bilgi verir. Bir modelin temsil ettiği gerçek dünyanın bölümüne, sistem, nesne sistemi, hitap edilen uzay, hedef sistem, incelenen sistem gibi isimler verilebilmektedir. Bu tanımda önemli olan modelin temsil ettiği gerçek dünya bölümünün sınırlarının belirlenmiş olmasıdır. Modeller, doğal dille, biçimsel ya da biçimsel olmayan, görsel ya da metinsel dillerle ifade edilebilirler. Modeller, gerçek dünyanın bir bölümünün soyutlanması sonucu ortaya çıkar. Soyutlama sonucunda modellenen gerçek dünya bölümünün bazı özellikleri modele yansıtılacak, bazı özellikleri ise ihmal edilecektir. Hangi özelliklerin modele yansıtılacağı, hangi özelliklerin ise ihmal edileceği

modelin oluşturulma amacı tarafından belirlenir. Bir modelin metamodeli ise, o model oluşturulurken kullanılabilecek yapı elemanlarını ve bu yapı elemanlarının kullanım kurallarını tanımlar. Bir model, metamodelinde tanımlı olan kurallara uymak zorundadır. Dolayısıyla metamodel "modelleme dilinin bir modeli" olarak tanımlanabilir.

Modelleri, yazılım geliştirme sürecinin odağına koyan yaklaşıma Model Güdümlü Mühendislik adı verilmektedir. Bu yaklaşıma Model Güdümlü Geliştirme adı da verilir. Model Güdümlü Geliştirme, OMG'nin ticari markasıdır.

Kurtev ve arkadaşları, Model Güdümlü Mimari yaklaşımını ve buna bağlı konuları açıklamak için üç kavramın irdelenmesinin önemine dikkat çekmektedir [8]. Bu kavramlar Şekil 3'te gösterilmiştir.



Şekil 3: İlkeler, standartlar ve araçlar.

Model Güdümlü Mimari ilkeleri, yaklaşımın kavramsal temelini oluşturur. Bu ilkeler farklı şekillerde uygulanarak farklı Model Güdümlü Mühendislik yaklaşımları ve farklı standartlar ortaya çıkarılabilir. Her Model Güdümlü Mimari yaklaşımı ve standardı, bunlara göre üretilen araçlarla hayata geçirilir.

Model Güdümlü Mühendislik yaklaşımının en çok bilinen iki gerçekleştirimi Object Management Group (OMG) tarafından 2001 yılında duyurulan "Model Güdümlü Mimari" ve Microsoft'un "Yazılım Fabrikaları"dır.

Model Güdümlü Mühendislik yaklaşımının amaçları, yazılım kalitesini arttırmak, yeniden kullanımı teşvik ederek yazılım geliştirme verimliliğini arttırmak, yazılım geliştirme maliyetlerini azaltmak, pazar değişimlerine daha hızlı cevap verebilme olanağını sağlamak, yazılım geliştirme sürecindeki karmaşıklıkları mümkün olduğu kadar azaltmak olarak özetlenebilir. Bu amaçların birbiriyle kesiştiği ya da birbirlerini tamamladıkları noktalar vardır.

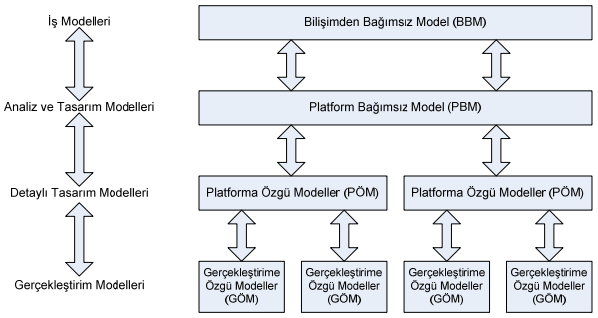
3.1. Model Güdümlü Mimari

Model Güdümlü Mimari, modelleri, yazılım geliştirme sürecinin merkezine yerleştirerek yazılım geliştirme süreçlerinin soyutlama seviyesini yükseltme hedefini bir adım daha ileri götürmüştür [9].

Model Güdümlü Mimari yaklaşımının temelini oluşturan maddeler şu şekilde özetlenebilir [10]:

- Modeller iyi tanımlanmış gösterimlerle ifade edilirler.
- Yazılım geliştirme süreci, temel olarak, bu modellerin, dönüşüm kurallarıyla başka modellere dönüştürülmesiyle, sonuçta hedeflenen ürünün elde edilmesi şeklinde ifade edilebilir.
- Modeller arasındaki dönüşümler için, bu modelleri biçimsel olarak tanımlayan metamodellere gereksinim vardır. Metamodeller, modeller arasındaki dönüşümleri ve modellerin birbirleriyle bütünleştirilmesini sağlar.
- Model Güdümlü Mimari yaklaşımının kabul edilmesi ve benimsenmesi için standartlar gereklidir.

OMG, Model Güdümlü Mimari yaklaşımı kapsamında, Şekil 4'te gösterildiği gibi katmanlar ve dönüşümler tanımlamıştır [10].



Şekil 4: Model güdümlü mimarinin katmanları ve dönüşümleri.

Şekil 4'teki platform kavramı, tutarlı bir işlev kümesini, arayüzler ve tanımlı kullanım desenleri ile sağlayan bir altsistem veya teknoloji kümesidir. Bir platformu kullanan sistemler, bu platformun sağladığı işlevleri, bu işlevlerin gerçekleştirim detaylarıyla ilgilenmeden kullanabilirler [11]. Örneğin, platform, bazı durumlarda bir işletim sistemi ve hizmet programları, bazı durumlarda J2EE veya .NET gibi iyi tanımlanmış programlama modeliyle temsil edilen bir teknoloji altyapısı olabilir [10].

Bilişimden bağımsız model, sistemin ne yapması gerektiğini, sistemin nasıl gerçekleştirileceği detaylarından bağımsız olarak tanımlar. Modelde kullanılan terminoloji, üzerinde çalışılan konu alanına ait, teknik olmayan terimlerden oluşur. Böylece bu model, iş analistleri gibi teknik bilgisi olmayan kişiler tarafından oluşturulabilir. Bilişimden bağımsız modelin amacı, teknik bilgileri olmayan alan uzmanları ile teknik kişilerin arasındaki boşluğu doldurmaktır. Bilişimden bağımsız model, geliştirilecek sistemle ilgili teknik bir bilgi içermediği için, bu modelin temsil ettiği bir sistem, mutlaka bilgisayar yazılımı olarak gerçekleştirilmek zorunda değildir. Aynı model mekanik bir sistemin geliştirilmesi için de kullanılabilir [12].

Platform bağımsız model, geliştirilecek sistemin üzerinde çalışacağı platformun teknik detaylarını içermez. Bunun için bu model, geliştirilen sisteme hizmet veren bir katmanın olduğu varsayılarak geliştirilir. Bu aşamada elde edilen modelin bir yazılım olarak gerçekleştirileceği bellidir fakat bu yazılımın hangi platform üzerinde çalışacağı belli değildir.

Platforma özgü model, platform bağımsız modelin gereksinim duyduğu hizmetlerin nasıl geliştirileceğini detaylı olarak tanımlar. Bu aşamada geliştirilecek yazılımın hangi platform üzerinde çalışacağı bellidir. Platforma özgü model hem iş seviyesinde hem de platform teknolojisi seviyesinde sistemi tanımlar ve kaynak kod için –gerçekleştirime özgü modeller- bir taban hazırlar.

Model Güdümlü Mimari, OMG standartlarından ve pratiklerinden oluşur. Model Güdümlü Mimari yaklaşımının kullandığı belirtimler aşağıdaki gibidir:

- Meta-Object Facility (MOF)
- Unified Modeling Language (UML)

- Common Warehouse Metamodel (CWM)

MOF, bir metadata yönetim çerçevesi ve model ve metadata güdümlü sistemlerin geliştirilmesini ve birlikte işlerliğini sağlamak için bir metadata servis kümesi sunar. Modelleme ve geliştirme araçları, veri ambarları, metadata havuzları, MOF'u kullanan sistemlere örnek olarak verilebilir. OMG'nin standartlaştırdığı UML, CWM, XMI (XML Metadata Interchange) gibi teknolojiler, metadata güdümlü değiştokuşu ve metadata işlemleri için MOF ve MOF tabanlı teknolojileri kullanırlar [13]. CWM'nin temel amacı, veri ambarı ve iş zekası metadatasının veri ambarı araçları, platformları ve metadata havuzları arasındaki değiştokuşu sağlamaktır [14]. CWM, UML, MOF ve XMI standartları üzerine temellendirilmiştir. UML, sistemlerin tanımlanması, geliştirilmesi ve belgelenmesi için kullanılan görsel bir dildir. UML, bütün uygulama alanlarında (finans, telekomünikasyon, sağlık, vs) ve gerçekleştirim platformlarında (J2EE, .NET) kullanılabilecek genel amaçlı bir modelleme dilidir [15].

3.2. Yazılım Fabrikaları

Yazılım Fabrikaları [16], Microsoft'un model güdümlü mühendislik ilkeleri çerçevesinde tanıttığı yaklaşımdır. Yazılım Fabrikaları, Microsoft'un yazılım mühendisliğinde kritik yenilikler olarak nitelediği pratikleri bünyesinde barındırır. Greenfield ve arkadaşları bu yenilikleri, sistemli yeniden kullanım, model güdümlü geliştirme, değişik bileşenleri birleştirerek geliştirme, süreç çerçeveleri ana başlıkları altında toplamışlardır [16].

Yazılım Fabrikaları, yeniden kullanımın sistemli bir şekilde yapılmasını öngördüğü için yazılım ürün hattı yaklaşımını benimsemiştir. Yazılım ürün hattı mühendisliği, yazılım üretiminin, kitlesel kişiselleştirme ilkeleri çerçevesinde yapılmasını önerir. Kitlesel kişiselleştirme kavramı, mal ve hizmetlerin, düşük miktarlarda da kitlesel üretimdeki kadar hızlı ve düşük maliyetli olarak üretilebilmesi anlamına gelmektedir. Yazılım üretiminde kitlesel kişiselleştirme, yazılım ürün ailelerindeki ürünlerin ortak ve farklı özelliklerinin belirlenmesiyle ve ortak özellikler için sistemli yeniden kullanım odaklı geliştirme yapılmasıyla mümkün olmaktadır.

Yazılım Fabrikaları, modellerin, soyutlama seviyesini yükseltmek için iyi bir araç olduğunu ve yazılım geliştirme sürecinin ana yapısı olarak kullanılması gerektiğini kabul eder. Modellerin kullanımıyla birlikte yazılım geliştirmede otomasyonu arttırmaya çalışır. Greenfield ve arkadaşlarına göre modeller, iş analistleri, konu alanı uzmanları, kullanıcılar gibi teknik bilgisi olmayan kişilerin anlayacağı terminolojiyi kullanarak soyutlama seviyesini yükseltmektedirler [16]. Bundan dolayı modelleme dilinin aşağıdaki özelliklere sahip olması gerektiğini belirtirler:

- Modelleme dilinin ne için tasarlandığı açıkça belirtilmelidir. Böylece konu alanı hakkında bilgi sahibi olan bir kişi dili değerlendirerek gereksinimlerine uygunluğunu sınayabilir.
- Modelleme dilindeki kavramlar, dilin kullanıldığı konu alanındaki kişiler tarafından kolayca anlaşılabilmelidir ve kullanılabilmelidir.
- Modelleme dilinin iyi tanımlanmış bir dil bilgisi olmalıdır.
- Modelleme dilindeki ifadelerin anlamları biçimsel olarak tanımlanmalıdır.

Yazılım Fabrikaları, yukarıdaki gereksinimleri karşılayan "Alana Özgü Diller"in modelleme için kullanılmasını önerir.

Yazılım fabrikalarının ana amacı yazılım geliştirmede verimliliğin ve performansın arttırılması ve yazılım geliştirme maliyetlerinin azaltılmasıdır. Model Güdümlü Mimari'nin aksine, yazılım fabrikaları, taşınabilirlik ve platform bağımsızlığı ile ilgilenmez. Bu hedeflere, performans, ölçeklenebilirlik ve güvenlik problemlerinden dolayı erişilemeyeceğini savunur.

4. Yazılım Mühendisliği ve Ontolojiler

Yapay zeka çalışma alanının uzun zamandır kullandığı ve üzerinde çalıştığı ontoloji kavramı, anlamsal web alanındaki gelişmeler nedeniyle, son yıllarda akademik dünyanın ve endüstrinin dikkatini çekmeye başlamıştır. Bunun sonucunda ontolojiler bir çok çalışma alanında kullanılmaya başlamıştır. Yazılım mühendisliği de ontolojilerin kullanıldığı alanlardan birisidir.

Biçimsel anlam temsilinin yazılım mühendisliği alanında kullanım fikri yeni değildir. Tetlow ve arkadaşları, çeşitli yapay zeka yaklaşımlarının, yazılım mühendisliğindeki uygulamalarına örnekler vermişlerdir [1].

Ontolojiler, yazılım mühendisliğinde farklı şekillerde ve farklı amaçlar için kullanılmaktadır. Happel ve Seedorf, ontolojilerin, yazılım mühendisliğindeki mevcut ve potansiyel uygulama alanlarını sınıflandırmışlardır [2].

Sınıflandırmanın bir boyutu ontolojilerin, yazılımın çalışma zamanında ya da geliştirme zamanında kullanımı değerlendirilerek oluşturulmuştur. Diğer boyutu ise ontolojinin yazılımın problem alanını ya da geliştirme ortamının altyapısını modellemesine göre belirlenmektedir. Bu boyutta ontolojilerin modellediği alan, yazılımın gerçek dünyadaki problem alanı ya da yazılım geliştirme sürecindeki çeşitli altyapı alanları (yazılım bileşenleri kütüphanesinin ontolojiler ile modellenmesi gibi) olabilmektedir. Bu iki boyuta göre Happel ve Seedorf, ontolojilerin yazılım mühendisliğindeki kullanımları için dört sınıf belirlemiştir:

- Ontoloji güdümlü geliştirme, ontolojilerin, yazılımın problem alanını tanımladığı ve gerçekleştirim zamanında kullanıldığı sınıfı temsil etmektedir. Bu kullanım sınıfına örnek olarak alan modellerinin ontolojilerle temsil edilmesi, ontoloji modelleme dillerinin diğer modelleme dilleriyle bütünleştirilmesi verilebilir.
- Ontoloji destekli geliştirmede, ontolojilerin gerçekleştirim zamanında ve yazılım geliştirme sürecinin iyileştirilmesinde kullanımını öngörmektedir. Bu kullanım sınıfına örnek olarak bileşenlerin yeniden kullanımı, kodlama desteği ve test verilebilir. Bu kullanım sınıfında ontolojiler yazılım geliştirme süreci kapsamındaki alanları modellemektedirler.
- Ontoloji tabanlı mimaride ise ontolojiler çalışma zamanında kullanılmakta ve uygulama mantığının temelini oluşturmaktadır. İş mantığının ontoloji destekli yapılarla temsil edilmesi bu uygulama sınıfı için bir örnek teşkil edebilir.
- Ontoloji destekli mimaride, ontolojiler, yazılıma çalışma zamanında altyapı desteği verirler. Ontolojilerin, web

servisleri tanımları üzerinde anlamsal bir katman oluşturarak, otomatik web servisi bulma, eşleştirme işlevselliklerini geliştirmesi bu kullanım sınıfı için bir örnektir.

Bu sınıflar arasındaki çizgiler çok keskin değildir. Kimi kullanımlar birden fazla sınıfa da dahil olabilir. Bu makale kapsamında ontoloji güdümlü geliştirme sınıfındaki uygulamalar incelenecektir.

Bilişim dünyasında, akademisyenlerin, endüstrideki araştırmacıların ya da bu iki alandan gelen kişilerin oluşturdukları grupların ait oldukları çok sayıda araştırma alanı; bu çalışma alanı kapsamında üzerinde çalıştıkları çok sayıda problem ve bu problemler için üretilmiş çözümler bulunmaktadır. Bu araştırma alanlarında üretilen çözümlerin, birbirleriyle ortak, çelişen ya da birbirini tamamlayıcı yönlerinin olması kaçınılmazdır. Bilişim dünyasına sunulan çözümlerin belirli ölçütlere göre sınıflandırılması bu çözümlerin ortak, çelişen ya da tamamlayıcı özelliklerinin bulunmasında ve bunların daha kolay ifade edilebilmesine olanak sağlayacaktır. Kurtev ve arkadaşlarının ortaya attığı "teknolojik uzay" kavramı bu amaca hizmet etmektedir [17].

Teknolojik uzay, birbiriyle bağlantılı kavramların, bilgi birikiminin, araçların, gerekli yeteneklerin ve imkanların oluşturduğu bir çalışma bağlamı olarak tanımlanmıştır [17]. Bir teknolojik uzay genellikle, ortak bir bilgi birikimine, eğitime, yazına ve hatta çalıştay ve konferanslara sahip bir kullanıcı topluluğuyla ilişkilendirilir. Teknolojik uzaylar birbirinden yalıtılmış değildir. Teknolojik uzaylar arasında çeşitli bağlantılar bulunabilir.

4.1. Model Güdümlü Mimari Tabanlı Ontoloji Mühendisliği

Model güdümlü mimari tabanlı ontoloji mühendisliği, Model Güdümlü Mimari ile Ontoloji Mühendisliği teknolojik uzaylarının birbirlerini tamamlayıcı özelliklerinden yola çıkılarak ortaya atılmış bir kavramdır.

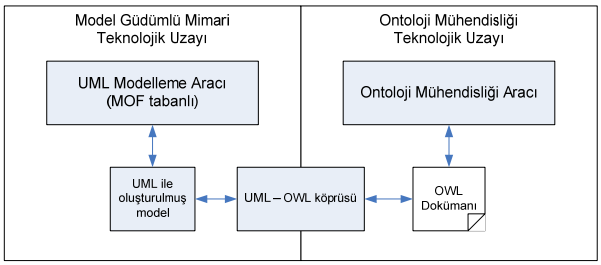
Ontoloji mühendisliği için geliştirilen araçlar bazı konulardaki yetersizlikleri için eleştirilmektedir [18,19,20]. Bu konular aşağıdaki başlıklar altında toplanabilir.

- Oluşturulan ontolojilerin fiziksel yönetimi ve araçlar arasındaki değiştokuşu
- Karmaşık ve büyük ölçekli ontolojilerin geliştirilmesi için gerekli olan ölçeklenebilirlik
- Çoklu kullanıcı desteği
- Yapılanış ve sürüm yönetimi

Yazılım mühendisliği alanında ve MOF'la birlikte, Model Güdümlü Mimari teknolojik uzayındaki stardartlar ve bu standartlara göre geliştirilmiş araçlar, ontoloji mühendisliği araçlarının çözüm getirmediği gereksinimleri karşılamışlar, bu gereksinimlere yönelik çözümler geliştirmişlerdir. Bu durumdan yola çıkılarak, ontoloji mühendisliği araçlarının bu eksiklerini giderebilmek için Model Güdümlü Mimari teknolojik uzayındaki standartların ve araçların kullanımı önerilmiştir [20].

Şekil 5'te MOF tabanlı UML modelleme araçları ile geliştirilmiş UML modellerinin, OWL dokümanına dönüştürülerek ontoloji mühendisliği araçlarına aktarımı gösterilmiştir. Benzer şekilde ontoloji mühendisliği araçlarıyla geliştirilmiş ontolojiler de MOF tabanlı modelleme araçlarına aktarılabilir. Bu şekilde, ontolojiler geliştirilirken, MOF tabanlı modelleme araçlarının sağladığı işlevler kullanılabilir. Bu yaklaşım aynı zamanda, günümüzdeki araştırma konularından birisi olan, ontolojilerin yazılımla nasıl bütünleştirilebileceği probleminin çözümü için de kullanılabilir.

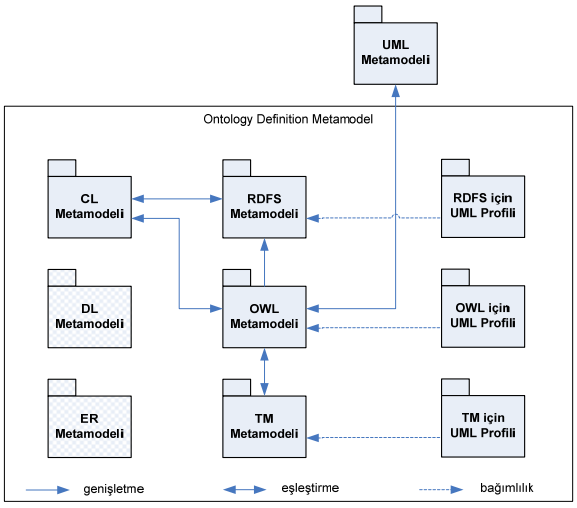
Şekil 5'te gösterilen UML-OWL köprüsü için OMG, Ontology Definition Metamodel (ODM) belirtimini duyurmuştur.



Şekil 5: MOF tabanlı araçlar ile ontoloji mühendisliği araçları arasında model değiştokuşu.

4.1.1. Ontology Definition Metamodel (ODM)

ODM belirtimi, üç temel bileşenden oluşur. Bunlar, Şekil 6'da gösterildiği gibi, metamodeller, matemodel eşleştirmeleri ve UML profilleridir [21].



Şekil 6: ODM'yi oluşturan bileşenler.

ODM, altı metamodel içermektedir. Bunlardan dördü normatif ikisi ise bilgilendiricidir [22]. ODM'ye uygun olduğunu iddia eden araçlar, normatif metamodellerle uyumlu olmak zorundadır. Bilgilendirici metamodeller ise standardın kavranmasına yardımcı olmak için kullanılan açıklamalardır. Bilgilendirici metamodeller, standartlara uygunluk açısından zorlayıcı değillerdir. ODM'nin içerdiği altı metamodel üç grup altında incelenebilir.

Common Logic (CL) ve Description Logic (DL), biçimsel mantık dillerini temsil ederler. CL metamodeli normatif, DL metamodeli ise bilgilendiricidir. Bu iki metamodel çok karmaşık bilgi temsiline olanak sağlar.

Resource Description Framework Schema (RDFS), Web Ontology Language (OWL) ve Topic Maps (TM) metamodelleri mantık dillerine göre daha az temsil gücüne sahip olan RDFS, OWL ve TM dillerinin soyut sözdizimlerini içerirler. Bu üç metamodel de normatiftir. OWL dili, RDFS'i temel aldığı için OWL metamodeli de, RDFS metamodeli genişletilerek oluşturulmuştur.

UML, yazılım mühendisliğinde kabul görmüş ve yoğun olarak kullanılan bir dildir. Bundan dolayı ODM belirtiminin içinde, zaten bir OMG standardı olan UML diliyle de bir bağlantı kurulmuştur. Bunun yanında yine kavramsal modelleme için çok kullanılan Entity Relationship (ER) diyagramları için de bilgilendirici bir matemodel oluşturulması planlanmıştır. Fakat uygulamada ER diyagramlarının standart bir kullanımı olmadığı için bu metamodelin gelecek nesil CWM standardının içinde oluşturulması daha uygun görülmüştür. Mevcut ODM belirtiminde yaygın olarak kullanılan ER ile UML yapı elemanları arasındaki ilişkiler gösterilmiştir.

RDFS, OWL ve TM için UML profilleri, UML dilinin ve araçlarının ontoloji modelleme için kullanımınını sağlar.

ODM'nin içerdiği metamodeller arasında eşleştirmeler de tanımlanmıştır. CL metamodeli ile RDFS ve OWL metamodelleri arasında, OWL metamodeli ile TM metamodeli arasında ve OWL metamodeli ile UML metamodeli arasında eşleştirmeler tanımlanmıştır. Ayrıca ER ile UML yapı elemanları arasındaki ilişkiler için bilgilendirici bir açıklama da ODM belirtimi içinde bulunmaktadır.

ODM, Model Güdümlü Mimari tabanlı ontoloji mühendisliğini bir çok yönde destekleyecek şekilde tasarlanmıştır [21]:

- İleri mühendislik: Ontolojiler, MOF tabanlı modelleme araçları (UML ile modelleme imkanı sunan araçlar) kullanılarak geliştirilebilecektir.
- Tersine mühendislik: Mevcut, anlamsal web ontoloji dilleriyle (OWL) geliştirilmiş olan ontolojilerin MOF tabanlı araçlara aktarılarak bu ontolojiler üzerinde işlemler yapılabilabilecektir.
- Tümleşik ontoloji/yazılım geliştirimi: Ontoloji geliştirme işlemi, yazılım geliştirmenin bir parçası haline getirilebilecektir.

Atkinson, OMG'nin ODM belirtiminin gereksiz olduğunu, UML'nin ve OWL gibi bilgi temsil dillerinin birleştirilerek tek bir dil yaratılmasının daha doğru olacağını savunmaktadır. UML'nin sistem modeli, ontoloji dillerinin ise bilgi modeli üzerine yoğunlaştığı, fakat sistem modeli ile bilgi modeli oluşturmanın aslında birbirinden farklı olmadığı fikrinin ise tek dil oluşturma düşüncesine temel oluşturduğunu vurgulamaktadır [23].

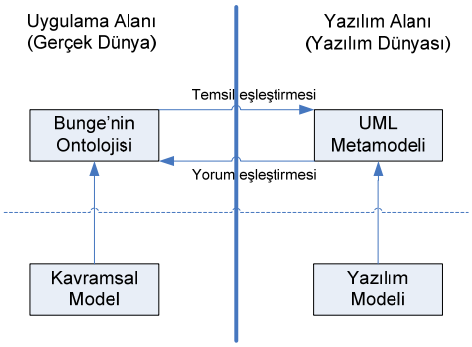
4.2. Gereksinim Mühendisliğinde Ontolojilerin Kullanımı

Bir sistemin, sunması gereken hizmetleri ve bu hizmetleri hangi kısıtlar altında çalışırken sunacağının belirlenmesi sürecine gereksinim mühendisliği adı verilmektedir. Bu tanımdaki mühendislik sözcüğü, geliştirilecek sistemin tanımlanması için sistemli bir sürecin uygulanacağını belirtmek amacıyla kullanılır [24].

Ontolojiler, gereksinim mühendisliğinde temel olarak iki şekilde kullanılabilir. Ontolojiler, gereksinim mühendisliği sürecini ya da problem alanını tanımlamak için kullanılabilir. Bu bildiri kapsamında, daha önce belirtildiği gibi ontolojilerin problem alanını tanımladığı durum ele alınacaktır.

Ontolojilerin, geliştirilecek sistemin gereksinimlerinin biçimsel temsilinde kullanılmasının, otomatik doğrulama, tutarlılık kontrolü gibi faydalar sağlayabileceği ve model güdümlü yazılım geliştirme yaklaşımlarının hayata geçirilebilmesi için bir ön koşul olacağı belirtilmektedir [2]. Bu düşünceler de ontolojilerin yazılım mühendisliği ile bütünleştirilmesi ihtiyacını destekleyen bir örnek olarak değerlendirilebilir.

Ontolojilerin gereksinim mühendisliğinde kullanımı için araştırmacılar çeşitli çalışmalar yapmıştır. Everman ve Wand, mevcut kavramsal modelleme yöntemlerinin, uygulama alanının biçimsel olarak tanımlanması için yeterli olmadığını belirtmektedir [25]. Yazılım modellemede kullanılan UML gibi dillerin yapı elemanlarının (sınıf, nesne gibi) anlamlarının, iş analistleri gibi teknik olmayan kişiler tarafından anlaşılması zordur. Bundan dolayı Everman ve Wand, nesneye yönelik yazılım modelleme dillerinin yapı elemanlarının anlamlarını ontoloji tabanlı olarak tanımlayı önermişlerdir. Everman ve Wand'ın yaklaşımı Şekil 7'de gösterilmiştir.



Şekil 7 : UML'nin yapı elemanlarının anlamının Bunge Ontolojisi kullanılarak tanımlanması.

Everman ve Wand, yaklaşımlarına bir örnek oluşturması için, UML dilinin yapı elemanlarının anlamlarını, Bunge'nin ontolojisini kullanarak tanımlamışlardır. Bir dilin anlamının ontoloji tabanlı olarak tanımlanması için iki tip eşleştirme gereklidir:

- Gerçek dünyadaki kavramların (ontolojideki kavramlar), dilde nasıl temsil edileceğini belirten temsil eşleştirmesi
- Dildeki yapı elemanlarının, gerçek dünyada nasıl yorumlanacağını belirten yorum eşleştirmesi

Everman ve Wand'ın yaklaşımı sayesinde, yazılım modelleme dilleri, iş analistleri gibi teknik olmayan kişiler tarafından kavramsal modelleme için kullanılabilecektir. Bunun yanında yazılım geliştirme sürecinde, kavramsal modellerden yazılım modellerine geçiş daha kolay sağlanabilecektir.

Bir başka çalışmada ise Lee ve Gandhi, gereksinim mühendisliği için Onto-ActRE çerçevesini önermişlerdir [26]. Bu çerçeve, değişik gereksinim modelleme teknikleriyle (Amaç güdümlü senaryo oluşturma, gereksinim alan modeli, bakış açısı sıradüzeni gibi) toplanan verilerin ontoloji mühendisliği sürecinden geçirilmesini önermektedir. Bunun sonucunda, sistemi farklı açılardan tanımlayan modeller arasındaki anlamsal farklılıklar ortadan kaldırılarak ve değişik

modeller arasındaki ilişkiler kurularak tümleştirilmiş bir gereksinim tanımı ortaya çıkarılması hedeflenmektedir.

5. Yaklaşım Farklılıkları

Model Güdümlü Mimari teknolojik uzayı ile ontoloji mühendisliği teknolojik uzayları farklı topluluklar tarafından farklı amaçlar için geliştirilmiştir. Bundan dolayı Model Güdümlü Mimari teknolojik uzayında bulunan standartlar ve yaklaşımlar ile ontoloji mühendisliği teknolojik uzayında bulunan standartlar ve yaklaşımlar arasında farklılıklar bulunmaktadır. Bu farklılıklar, bu bildiri kapsamında, katmanlı mimari, açık-kapalı dünya yaklaşımları ve birlikte işlerliğe yaklaşım başlıkları altında incelenmiştir.

5.1. Katmanlı Mimari

Model Güdümlü Mimari için OMG tarafından tanımlanmış ve genel olarak kabul görmüş dört katmanlı bir mimari bulunmaktadır. Fakat ontoloji dilleri için geniş kesimler tarafından kabul edilmiş bir katmanlı mimari bulunmamaktadır.

Daconta ve arkadaşları, ontolojiler için farklı temsil katmanlarının gerekliliğinin altını çizmişler ve ontoloji temsili için, ontoloji modelleme dilinden, alana özgü olarak geliştirilmiş modellerden ve bu ontoloji modeliyle uyumlu örneklerden oluşan üç katman önermişlerdir [27]. Bezivin ve arkadaşları, Model Güdümlü Mimari ve ontoloji mühendisliği teknolojik uzayları arasında bir köprü kurabilmek amacıyla ontoloji mühendisliği için dört katmanlı bir yapı önermişlerdir [28]. Pan ve Horrocks, RDFS için, Model Güdümlü Mimari'deki dört katmanlı yapıyla uyumlu bir mimari önermişlerdir [29]. Bu çalışmalardan çıkan sonuç, Model Güdümlü Mimari teknolojik uzayı ile ontoloji mühendisliği arasındaki köprünün kurulabilmesinin gerekliliklerinden biri, ontoloji mühendisliği için de Model Güdümlü Mimari'de olduğu gibi dört katmanlı standart bir mimari tanımlayarak katmanlar arasında eşleştirmeler oluşturmaktır.

5.2. Açık-Kapalı Dünya Yaklaşımları

Açık dünya yaklaşımında bir önermenin doğruluk değeri doğru, yanlış ya da bilinmiyor olabilir. Kapalı dünya yaklaşımında ise doğruluğu bilinmeyen ya da çıkarsanamayan bir önerme yanlıştır.

Anlamsal web vizyonunu gerçekleştirmek için geliştirilen ontoloji dilleri açık dünya yaklaşımını benimsemektedir. Bu dillerin, web gibi yoğun ve sürekli değişen veri barındıran, dağıtık ve birbiriyle çelişen verinin bulunabileceği bir ortamda kullanılmak için tasarlandığı düşünüldüğünde, açık dünya yaklaşımının bu koşullara çok daha uygun olduğu anlaşılacaktır. Diğer taraftan yazılım mühendisliğinde kullanılan UML gibi modelleme dilleri ise kapalı dünya yaklaşımını benimsemektedir.

Bir dilin, açık ya da kapalı dünya yaklaşımını benimsemesi, üretilen bir model üzerinde yapılacak bir çıkarsamada farklılıklar oluşturacaktır [30]. Örneğin, bir modelde her kişinin bir ismi olması gerektiği belirtilsin. UML ile oluşturulmuş bir modelde, bir kişinin isminin olmaması, her kişinin bir ismi olması gerektiği kuralını bozduğu için modelde bir hata oluşacaktır. Fakat aynı durum açık dünya yaklaşımını uygulayan bir dille temsil edilen bir modelde oluştuğunda bir kişinin isminin olmaması durumu, o kişinin isminin bilinmediği şeklinde yorumlanacaktır.

Ontoloji dillerinin de bazı durumlar için kapalı dünya yaklaşımını da desteklemesinin faydalı olacağı da belirtilmiştir [31, 32].

5.3. Birlikte İşlerliğe Yaklaşım

Model Güdümlü Mimari, modellerin fiziksel olarak saklanması, değiştokuşu ve yönetimi üzerine yoğunlaşmaktadır. Dolayısıyla Model Güdümlü Mimari, sistemler arasında birlikte işlerlik için ancak sözdizimsel seviyede bir dönüşüm için temel hazırlayabilmektedir. Ontoloji mühendisliği teknolojik uzayındaki diller ise anlamın biçimsel temsiline yoğunlaştıkları için anlamsal seviyede bir birlikte işlerlik için temel hazırlayabilmektedir.

Atkinson ve Kühne Model Güdümlü Mimari'deki farklı meta-katmanlardaki elemanlar arasındaki ilişkileri irdelemişlerdir [33]. Model Güdümlü Mimari'de farklı katmanlardaki elemanların arasındaki ilişkilerle aynı katmanda bulunan elemanların arasındaki ilişkileri ayrıştırmışlardır. Model Güdümlü Mimari'nin dört katmanındaki elemanların arasındaki ilişkilerin dilbilimsel, aynı katmandaki elemanların arasındaki ilişkileri de ontolojik olarak adlandırmışladır. Böylece Model Güdümlü Mimari'deki metamodelleme kavramının da dilbilimsel olarak nitelendirilmesi gerektiğini ve bu mimaride bir de ontolojik metamodelleme kavramının olması gerektiğini savunmaktadırlar. Ontolojik metamodellemenin Model Güdümlü Mimari teknolojik uzayı içerisinde irdelenmesi, anlamsal seviyede birlikte işlerlik için yeni yaklaşımlar doğurabilecektir.

6. Sonuçlar

Yazılım mühendisliği, modellerin, yazılım geliştirme süreçlerinin odağına yerleştirildiği yaklaşımlara doğru yönelmektedir. Modellerin, sadece taslak tasarımlarda ve belgeleme amaçlı kullanımları, yeni yazılım geliştirme yaklaşımlarında terk edilerek, modellerin gereksinim analizi aşamasından başlanarak kullanılması önerilmektedir.

Model güdümlü mühendislik yaklaşımının, farklı kurumlar tarafından sunulan gerçekleştirimlerinde önemli farklılıklar göze çarpsa da, biçimsel bilgi temsiline daha çok önem verilmesi gerekliliği, yaklaşımların ortak noktası olarak göze çarpmaktadır. Model Güdümlü Mimari'nin temelini oluşturan standartlardan birisi olan UML'nin anlamının biçimsel olarak tanımlanması çalışmaları da bu gözlemi destekler niteliktedir.

Ontolojilerin yazılım mühendisliğinde kullanımının faydalı olacağı konusunda araştırmacılar arasında genel bir fikir birliği görülmektedir. Ontolojilerin yazılım geliştirme süreçleriyle bütünleştirilmesi konusundaki çalışmalar genellikle, OMG'nin Model Güdümlü Mimari yaklaşımı kapsamında yapılmaktadır.

Bir konu alanını biçimsel olarak modelleyen ontolojilerin, yazılım geliştirme sürecinin gereksinim analizi safhasında kullanılması, bu safhadaki çalışmaları hızlandırarak maliyetleri düşürebilir. Ontolojilerin yeniden kullanımı destekleyici, genişletilebilir doğası, yazılım geliştirme süreçlerinde ontolojilerin kullanımını kolaylaştırıcı niteliktedir. Gereksinim analizinden başlayarak modellerin, biçimsel olarak temsil edilebilmesi ve bu modellerden diğer safhalardaki modellere geçişlerde, aralardaki bağlantıların korunabilmesi, model güdümlü yazılım geliştirme amacına ulaşma yolunda geçilmesi gereken önemli aşamalardan

biridir. Gereksinim analizinde oluşturulan modellerden başlayarak, daha alt seviyedeki modellere kadar izlenebilirliğin sağlanabilmesi, iş gereksinimlerindeki değişikliklerin, alt seviyedeki modelleri ve üretilen kodu nasıl etkileyebileceğinin tespit edilmesini sağlayabilir. Böylece bir gereksinim değişikliğinin, gerçekleştirilmeden önce yazılım üzerindeki etkileri ve yaklaşık maliyeti hesaplanabilir. Bu faydaların gerçeklenebileceği çözümler oluşturulurken, model güdümlü mimari ile ontoloji mühendisliği teknolojik uzayları arasındaki yaklaşım farklılıklarının göz önüne alınması gereklidir. Model güdümlü mimarinin genel kabul görmüş dört katmanlı mimarisine karşılık ontolojilerin bu katmanlı mimarideki yerinin net olarak belirlenememesi, ontolojilerin açık dünya yaklaşımını benimserken yazılım modelleme dillerinden en çok kabul gören UML'nin kapalı dünya yaklaşımına göre tasarlanmış olması ve ontolojilerin anlamsal seviyede birlikte işlerliğe vurgu yaparken model güdümlü mimarinin sözdizimsel seviyede dönüşümler için temel hazırlaması, ontolojilerin yazılım mühendisliğinde kullanımında göz önüne alınması gereken noktalar olarak karşımıza çıkmaktadır.

7. Kaynakça

[1] Tetlow, P., Pan, J.Z., Oberle, D., Wallace, E., Uschold, M., Kendall, E., Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering, 2006.

[2] Happel, H.-J., Seedorf, S., Applications of Ontologies in Software Engineering. International Workshop on Semantic Web Enabled Software Engineering (SWESE'06), Athens, USA, 2006.

[3] Gruber, T.R., A translation approach to portable ontology specifications, Knowl. Acquis. 5, 199-220, 1993.

[4] Uschold, M., Gruninger, M., Ontologies and semantics for seamless connectivity, SIGMOD Rec. 33, 58-64, 2004.

[5] Daconta, M.C., Smith, K.T., Obrst, L.J., The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management, John Wiley & Sons, Inc., 2003.

[6] Abran, A., Bourque, P., Dupuis, R., Moore, J., Tripp, L., Guide to the Software Engineering Body of Knowledge – SWEBOK, IEEE, 2004.

[7] Mellor, S., Scott, K., Uhl, A., Weise, D., MDA Distilled: Principles of Model-Driven Architecture, Addison-Wesley, 2004.

[8] Kurtev, I., Bezivin, J., Jouault, F., Valduriez, P., Model-based DSL frameworks, Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, ACM Press, Portland, Oregon, USA, 2006.

[9] Arlow, J., Neustadt, I.: Enterprise Patterns and MDA, Building Better Software with Archetype Patterns and UML. Addison Wesley Longman Publishing Co., Inc., 2003.

[10] Brown, A.W., Conallen, J., Tropeano, D., Introduction: Models, Modeling, and Model-Driven Architecture (MDA), 2005.

[11] Miller, J., Mukerji, J., MDA Guide Version 1.0.1, Object Management Group, 2003.

[12] Meservy, T. O., Fenstermacher, K. D., Transforming Software Development: An MDA Road Map, Computer, vol. 38, sayfa 52-58, 2005.

[13] Meta Object Facility (MOF) 2.0 Core Specification Version 2.0, Object Management Group, 2004.

[14] Common Warehouse Metamodel (CWM) Specification Version 1.1, Object Management Group, 2003.

[15] UML Specification: Infrastructure Version 2.0, Object Management Group, 2004.

[16] Greenfield, J., Short, K., Cook, S., Kent, S., Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Wiley 2004.

[17] Kurtev, I., Bézivin, J., Aksit, M., Technological Spaces: An Initial Appraisal. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002.

[18] Frankel, D., Hayes, P., Kendall, E., McGuinness, D., The Model Driven Semantic Web, 1st International Workshop on the Model-Driven Semantic Web (MDSW2004), Monterey, California, USA, 2004.

[19] Pan, Y., Xie, G., Ma, L., Yang, Y., Qiu, Z., Lee, J., An MDA-Based System for Ontology Engineering, 2005.

[20] Kendall, E., The Model Driven Semantic Web – Emerging Technologies & Implementation Strategies, http://ontolog.cim3.net/file/resource/presentation/ElisaKendall_20050908/The_Model_Driven_Semantic_Web--ElisaKendall_Recording-2079260-149405_20050908.mp3, 2005.

[21] Chang, D.T., Kendall, E., Major Influences on the Design of ODM, 1st International Workshop on the Model-Driven Semantic Web (MDSW2004), 2004.

[22] Ontology Definition Metamodel Sixth Revised Submission to OMG/RFP ad/2003-03-40, Object Management Group, 2006.

[23] Atkinson, C., Unifying MDA and Knowledge Representation Technologies, The Model-Driven Semantic Web Workshop (MDSW 2004), 2004.

[24] Sommerville, I., Software engineering (5th ed.), Addison Wesley Longman Publishing Co., Inc., 1995.

[25] Evermann, J., Wand, Y., Ontology based object-oriented domain modelling: fundamental concepts, Requir. Eng., vol. 10, sayfa 146-160, 2005.

[26] Lee, S. W. and Gandhi, R. A., Ontology-based Active Requirements Engineering Framework, in Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05) - Volume 00: IEEE Computer Society, 2005.

[27] Daconta, M.C., Smith, K.T., Obrst, L.J., The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management, John Wiley & Sons, Inc., 2003.

[28] Bézivin, J., Devedzic, V., Djuric, D., Favreau, J.M., Gasevic, D., Jouault, F., An M3-Neutral Infrastructure for Bridging Model Engineering and Ontology Engineering, 2006.

[29] Pan, J.Z., Horrocks, I., Metamodeling Architecture of Web Ontology Languages, SWWS, 2001.

[30] Baclawski, K., Kokar, M.M., Kogut, P.A., Hart, L., Smith, J.E., Letkowski, J., Emery, P., Extending the Unified Modeling Language for ontology development, Software and System Modeling 1, 142-156, 2002.

[31] Web Ontology Language Use Cases and Requirements, In: Heflin, J. (ed.), W3C, 2003.

[32] Antoniou, G., Harmelen, F.v., A Semantic Web Primer, The MIT Press, Cambridge, Massachusetts, 2004.

[33] Atkinson, C., Kühne, T., Model-Driven Development: A Metamodeling Foundation, IEEE Softw. 20, 36-41, 2003.