Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 19 Sayı 55.1 (Özel Sayı) Ocak 2017

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 19 Issue 55.1. (Special Issue) January 2017

DOI: 10.21205/deufmd.2017195527

Yazılım Yeniden Yapılamaya Yönelik Bir Kurumsal Mimari: Model Güdümlü ve Ontoloji Tabanlı Bir Yaklaşım

Murat Paşa UYSAL*[1], A. Erhan MERGEN[2]

[1]Ufuk Üniversitesi, İİBF, Yönetim Bilişim Sistemleri Bölümü, Ankara
[2]Rochester Institute of Technology, Saunders College of Business, Rochester, NY, USA

(Alınış / Received: 03.06.2016, Kabul / Accepted: 12.08.2016,
Online Yayınlanma / Published Online: 09.01.2017)

Anahtar Kelimeler
Yazılım Yeniden Yapılama,
Kurumsal Mimari,
Model Güdümlü Mimari,
Ontoloji

Özet: Tekrar kullanılabilirlik, bütünleştirme, anlamsal iletişim ve birlikte çalışabilirlik Yazılım Yeniden Yapılama (YYY) projelerinde karşılaşılabilen ana sorunlar arasındadır. Bu kapsamda çalışmamızda, YYY yönelik bir Kurumsal Mimari (KM) geliştirilmiş, ontolojik yöntemlerle test ve değerlendirilmiştir. Tasarım Bilimi Araştırma Yöntemi doğrultusunda yürütülen araştırmanın ana bileşenleri ve teorik temellerini YYY, Model Güdümlü Mimari, Kurumsal Mimari ve Ontoloji bilgi alanları oluşturmuştur. Çalışmanın yazılım mühendisliği alanına olan katkılarını, (a) YYY sürecine KM ile bütüncül yaklaşılması ile (b) KM ve YYY süreçlerinin anlamsal yapılarının ontolojik yöntemlerle iyileştirilmesi olarak göstermek mümkündür. İlk izlenimlerimiz, geliştirilen KM'nin değişik soyutlama düzeylerindeki YYY problemlerine farklı bakış açılarını kazandırarak yazılımla ilgili paydaşların görüş ve ihtiyaçlarını karşılayabileceği yönündedir.

An Enterprise Architecture for Software Re-Engineering: A Model-Driven and Ontology-Based Approach

Keywords
Software Reengineering,
Enterprise Architecture,
Model-Driven Architecture,
Ontology

Abstract: Reusability, semantic communication, interoperability may be the major problems during Software Re-engineering (SRE) projects. In this study, therefore, we design and develop a SRE Enterprise Architecture (EA) and evaluate it using ontological methods and techniques. The study is conducted according to the guidelines and principles of Design Science Research Method. The SRE, Model-Driven Architecture and Ontology knowledge domains formed the theoretical foundations of our research. The contributions of the study to Software Engineering Research Domain could be (a) the holistic and enterprise architectural approach adopted for SRE and (b) improving the sematic architecture of SRE processes using ontological evaluation methods and techniques. Our first impression is as it can provide different views to SRE issues at various abstraction levels while it can represent the requirements of various stakeholders in a SRE project.

*Sorumlu yazar: mpuysal@gmail.com

1. Giriş

Yazılım Yeniden Yapılama (YYY) (Software Re-engineering) ile ilgili çalışmalar incelendiğinde temel problem sahalarının, (a) eski (legacy) yazılım sistemlerinin kalitesinin iyileştirilmesi ile (b) işlevsel ve işlevsel olmayan özelliklerin geliştirilmesi olduğu gözlenmektedir [1]. Bu amaçla, YYY ile ilgili araştırmalar yapılmış ve süreçlerin iyileştirilmesine yönelik çeşitli yöntem, teknik ve araçlar önerilmiştir [2, 3]. Ancak, YYY yoğun kaynak ve zaman kullanımını gerektiren, gidiş-dönüşlü (round-trip), yinelemeli ve artırımsal yazılım mühendisliği etkinliklerini içeren bir süreçtir [4]. Dolayısıyla, YYY süreçleri otomatik hale getirilebilmeli, yazılım ürün ve araçları yeniden yapılanan yazılımla ilgili sonraki aşama ve süreçlerde de kullanılabilmelidir. Bu durum, çalışmamızın birinci araştırma problemini oluşturmaktadır. Bu kapsamda, Model Güdümlü Yazılım Geliştirme (MGYG) ve Model Güdümlü Mimari (MGM), yazılım mühendisliği süreçlerinin otomatik hale getirilmesini sağlayabilecek yaklaşım ve standartlar olarak gösterilebilirler [5]. Böylece, yazılımla ilgili kalite, etkililik ve öngörülebilirlik konuları farklı modelleme ve soyutlama düzeylerinde ele alınabilmektedir. MGYG'de sistemlerin anlaşılırlığı ile sürecin kolaylaştırılması amacıyla çeşitli modelleme yöntem, araç ve tekniklerin kullanılmaktadır [6-11].

Öte yandan, yazılım sistemlerinin ömür devri boyunca sadece yazılım teknolojileriyle ilgili eksiklik veya yeni ihtiyaçlar ortaya çıkmamaktadır. Aynı zamanda, ilgili kuruma ait değişen iş süreçleri, veri yönetimindeki değişiklikler ve yeni ihtiyaçlar ile donanımsal gereksinimler bir YYY projesi süresince karşılaşılabilecek diğer sorunlar arasındadır. İş süreci-veri-yazılım-altyapı boyutlarında YYY mimarisine bütüncül ve tümleşik yaklaşılmasını gerektiren söz konusu bu durum, araştırmamızda ele alınan bir diğer problemi teşkil etmektedir. Bu bağlamda, bilişim sistemlerinin tasarımında kullanılan Kurumsal Mimari (KM) (Enterprise Architecture) yöntem, teknik ve araçlarının, araştırma probleminin bu boyutundaki çözümüne katkıda bulunabileceği söylenebilir [12].

YYY araştırma alanı MGYG ve KM çerçevesinde incelendiğinde; (a) süreçler arasındaki bilgi yönetimi ve bilgi paylaşımı, (b) farklı nitelikte, mimaride ve platformdaki yazılımların entegrasyonu ve değerlendirilmesi, (c) bunlar arasındaki iletişim ve birlikte çalışabilirliğinin sağlanması vb. konularda yapısal, kavramsal ve anlamsal nitelikteki sorunlarla karşılaşıldığı gözlenmektedir. Dahası, bu yapılar arasındaki kavramsal bağımlılıkların, tutarlılıkların ve mimari yapıların, çeşitli standartlara ve belirtimlere uyumluluklarının kontrol edilebilmesi ve değerlendirilebilmesi gerekmektedir. MGYM sürecinde kullanılan program dillerindeki bileşenlerin işlem, mantık ve gösterime yönelik anlamsal yapıları genellikle açık olarak ifade edilememektedir. Kullanılan yazılım modellerinin anlamsal tutarlılıkları ise geleneksel veya prosedürel yöntem ve araçlarla gerçekleştirilmekte, bu durum yazılım sürecindeki modellerin ileriye ve tersine dönüştürülmesinde çeşitli güçlükleri ortaya koyabilmektedir.

Bu bağlamda, bilgi mühendisliği alanında kullanılan ontoloji yöntem, araç ve tekniklerinin, yazılım mimarisi ve KM'deki söz konusu yapısal ve anlamsal boyuttaki problemlerin çözümünde kullanılabileceği değerlendirilmektedir. Ancak literatürdeki çalışmalar incelendiğinde, YYY araştırma alanında MGYG ve Ontoloji kavramlarını birlikte ele alan çalışmaların sınırlı düzeyde olduğu, YYY ve KM'i içeren araştırma ve

deneyim çalışmalarının bulunmadığı gözlenmektedir. Dolayısıyla bu makalede, söz konusu araştırma problemlerine yönelik bir KM geliştirilerek ontolojik yöntemlerle analiz edilmiş ve değerlendirilmiştir. Çalışmamızın yazılım mühendisliği araştırma alanına olan katkılarını aşağıdaki gibi sıralamak mümkündür:

(a) YYY'da bulunan bütün süreçlere ve bileşenlere bütüncül ve tümleşik olarak yaklaşılması, bu amaçla YYY'ya yönelik bir KM'nin geliştirilmesi,
(b) MGYG, YYY bileşenleri ve önerilen KM modelinin anlamsal yapılarının, tasarım ve değerlendirme süreçlerinin, ontolojik yöntem, teknik ve araçlar kullanılarak iyileştirilmesidir.

Makalenin sonraki bölümlerini, çalışmanın teorik temelleri, araştırma yöntemi ve geliştirilen kurumsal mimariyi içeren başlıklar oluşturmaktadır.

2. Teorik Çerçeve

2.1. Yazılım Yeniden Yapılama

Yeniden Yapılamayı (re-engineering), yapılanacak bir sistemi, mevcut sistemle aynı ya da daha üst seviye bir soyutlama düzeyinde yeniden oluşturma ve yeni sistemi uygulamalarla devam ettirme olarak tanımlamak mümkündür [9]. Aynı kapsamda YYY süreci, (a) evrimleşebilen bir sistem oluşturmak, (b) mevcut yazılımın işlevlerini geliştirmek, (c) ona yeni işlevler katarak (d) kalitesini iyileştirmek amacıyla gerçekleştirilir. Bu süreç, (a) tersine mühendislik (reverse engineering), (b) yeniden düzenleme (restructuring, refactoring) ve (c) ileriye mühendislik (forward engineering) yöntemlerini içerebilmektedir. Bu kapsamda YYY, genel olarak program dönüştürme (program transformation) ve program gösterimi (program representation) işlemlerinden oluştuğu söylenebilir. Program dönüştürmede, çeşitli gereksinim ve ölçütlere (yazılım dili, hedef mimari, soyutlama düzeyleri vb.) bağlı olarak, program çevirisi (translation), yazılım göçü (migration), optimizasyonu vb. etkinlikler gerçekleşebilmektedir. Program gösteriminde ise ayrıştırma ağaçları (parse tree), soyut söz dizim ağaçları (abstract syntax tree), çizgeler (graph) vb gösterim yöntemlerinden biri veya birkaçı kullanılabilmektedir [2].

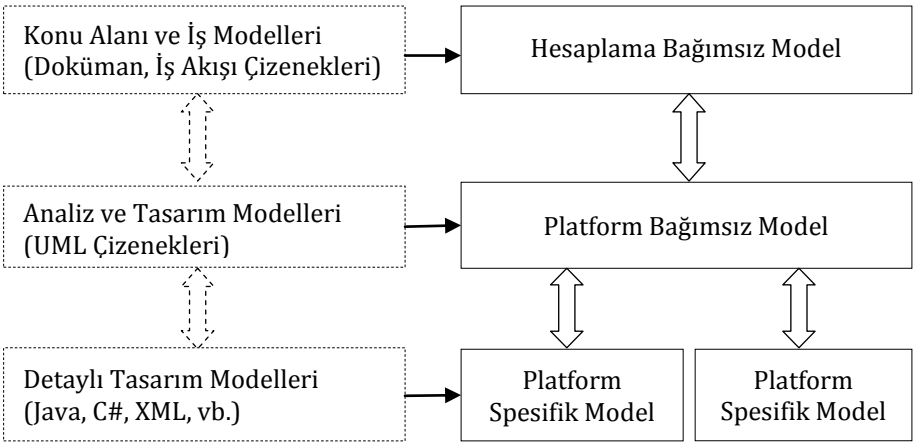
2.2. Model Güdümlü Mimari

Modelleme ve MGM, MGYG olarak bilinen yazılım geliştirme yaklaşımının temelini oluşturmaktadır [10]. Modeller bir problem alanıyla ilgili karar alma ve ona yönelik bir çözüm geliştirmek amacıyla kullanılırlar. Dolayısıyla, modeller ve aralarındaki ilişkiler, probleme yönelik çözümün yaratıldığı süreci kayıt altına alır ve karşılıklı bağımlılıkları içeren yapıyı da ortaya koyarlar. Bu ilişkiler aynı zamanda sistem tasarlama ve geliştirme sürecinin herhangi bir noktasındaki değişikliklerin anlaşılabilmesini, etkilerinin öngörülebilmesini sağlamaktadırlar. MGM'de yazılım geliştirme süreci, modeller arasında bir dizi dönüşüm olarak gerçekleşmekte, çeşitli katman ve dönüşüm işlemlerinden oluşan bir mimari çerçeve doğrultusunda evirilmektedir. MGYG'de temel amaçlardan birisi, yazılım karmaşıklığını gidermek amacıyla modeller aracılığıyla yazılım süreçlerinde genelleme ve soyutlama düzeylerinin artırılması, geliştirilen modeller ile yazılım kodlarına bir temel oluşturulmasıdır.

Şekil 1'de görüldüğü gibi MGM'de yazılım tasarımı ve geliştirilmesine üç farklı bakış açısıyla yaklaşılmaktadır. İlk aşamada, Hesaplama Bağımsız Modeller, teknolojiden bağımsız olarak sistemin nasıl gerçekleştirileceğini belirlemektedir. Platform Bağımsız Modeller, teknik detaylardan soyutlanmış ve yine platformdan

bağımsız yazılımla ilgili bir grup servisi tanımlamaktadırlar. Son olarak Platform Spesifik Modeller, yazılımın kullanılacağı hedef platformu dikkate almakta ve sisteme yönelik teknik detayları içermektedir. Üç farklı nitelikteki bu modeller, çeşitli kurallar doğrultusunda birbirlerine dönüştürülmektedirler. Dolayısıyla, yazılımın analiz, tasarım veya kodlama aşamasındaki herhangi bir değişiklik gidiş-dönüşlü olarak diğer aşamalara kolayca yansıtılabilmektedir.

2.3. Kurumsal Mimari

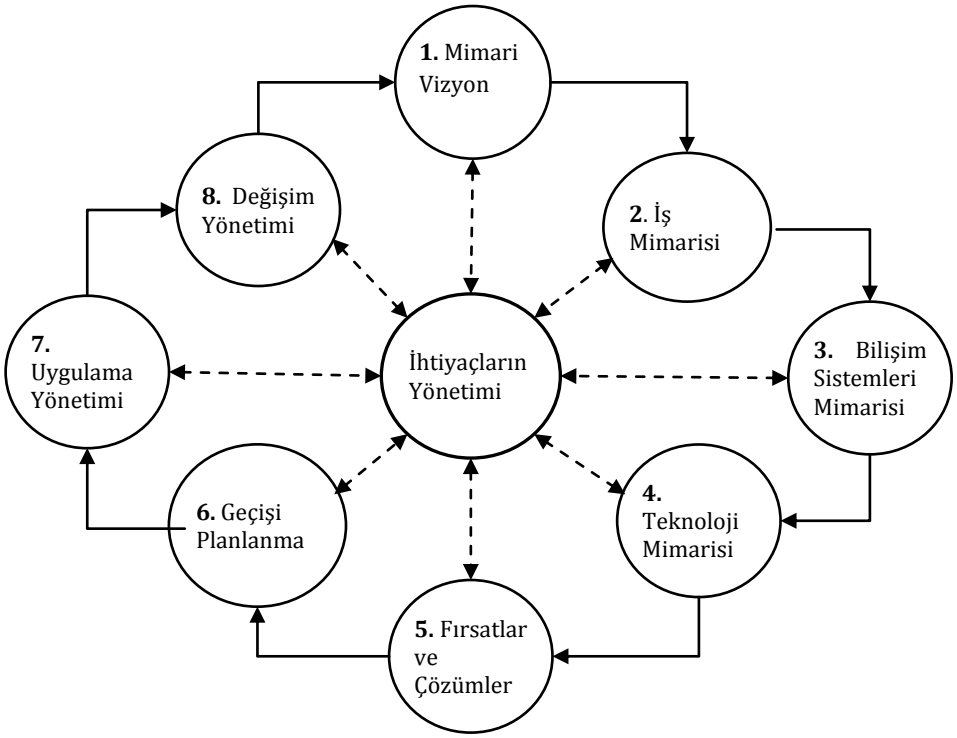


Şekil 1. Model Güdümlü Mimari ([14]'den alınmıştır)

KM kavramını, bir işletme ve kuruluşa ait yapıyı, süreçleri, bilişim sistemleri ile alt yapısının tasarımı ve geliştirilmesinde kullanılan, birbirleriyle tutarlı yöntemler, modeller, ilke ve prensiplerin bütünü olarak tanımlamak mümkündür [12]. Bir KM'nin en önemli özelliği süreç, veri, yazılım, donanım ve altyapı boyutunda farklı paydaş ve uzmanların ilgi sahasını da kapsayacak şekilde işletmeye yönelik bütüncül bir görünüm ve bakış açısını sunmasıdır. KM'ler, bilişim stratejileriyle iş stratejilerini, organizasyonun alt yapısı ve süreçleriyle bilişim teknolojileri alt yapısı ve süreçlerini uyumlu hale getirmekte ve kurumun BT stratejilerinin hayata geçirilmesine kolaylık sağlamaktadırlar.

TOGAF, Zachman, DoDAF, IBM EA ve UML'i belli başlı KM tasarım yaklaşımları veya endüstri uygulamaları olarak göstermek mümkündür [12]. Bu yaklaşımlar, çeşitli ölçütlere göre birbirleriyle karşılaştırıldığında (soyutlama düzeyleri, çıktılar, çerçeve, içerdikleri süreç modelleri ve KM geliştirme yöntemi vb.) aralarında bazı farklılıklar gözlenebilmektedir. Literatürdeki çalışmalar ve endüstrideki uygulamalar, kurumsal ihtiyaçlara göre bu yaklaşımlardan birisinin veya harmanlanmış ve bütünleşik bir biçiminin kullanılmasını önermektedir [12]. Ancak KM tasarım ve geliştirme konusu, YYY süreci, sistem geliştirme ömür devri ve MGYG ile birlikte ele alındığında, TOGAF v9.1 standardı ve içerdiği Mimari Geliştirme Yönteminin (Architecture Development Method) (ADM) bu araştırmanın amaçlarıyla daha uyumlu olduğu gözlenmektedir. Dolayısıyla çalışmamızda, araştırma problemlerin çözümüne yönelik olarak TOGAF kapsamındaki KM yaklaşımı benimsenmiştir.

TOGAF standardı, iş süreçleri, kurumsal veri, uygulama ve teknoloji olmak üzere işletmedeki dört ana mimari alanla ilgilenmekte ve onları desteklemektedir. Sunmuş olduğu Mimari Geliştirme Yöntemi ile Şekil 2'de gösterildiği gibi sekiz aşamalı ve döngüsel biçimde KM'ler geliştirilmektedir. Bu yöntemin kullanımı, tanıtımı ve bilimsel temelleri



Şekil 2. TOGAF Mimari Geliştirme Yöntemi (ADM) [13]

ayrı bir çalışma konusu olup ayrıntılar okuyucuya bırakılmıştır.

2.4. Ontoloji

Ontolojiyi bir kavram alanına yönelik açık belirtimler veya üzerinde anlaşılan, paylaşılan bir kavram kümesi olarak tanımlanmak mümkündür [20]. Bu bağlamda bir ontoloji, herhangi bir konu alanıyla ilgili temel terim ve kavramları, onlar arasındaki ilişkileri, bu kavram ve ilişkiler arasındaki kuralları içermektedir. Ontolojiler, aynı zamanda bir bilgi tabanının temelini oluşturmak üzere ilgili konu alanını tanımlayan ve çeşitli biçimlerde yapılandırılan terimler, kavramlar ve ilişkiler topluluğudur. Bu kapsamda ontolojileri, çeşitli bilgi gösterim ve modelleme bileşenlerinin (kavram, sınıf, ilişki vb.) kullanıldığı, sahip olduğu bilgi temsil gücüne de bağlı olarak, alt seviye (yazılım mühendisliği, veritabanı tasarımları) ya da üst seviye (tanımlayıcı mantık, yapay zekâ çerçeveleri) ontolojiler biçiminde tasnif etmek de mümkündür. Ontolojiler yapay zekâ, yazılım mühendisliği ve veritabanı teknikleri kullanılarak çeşitli amaçlar için uygulamaya konulabilmektedir.

Ontolojisinin bir başka uygulama biçimi olan Anlamsal Web uygulamalarında, bilginin biçimsel olarak temsil edilmesi ile bu bilginin bilgi sistemleri aracılığıyla işlenebilmesi hedeflenmektedir. Ontolojiler, bilgiyi temsil gücüne ve anlamsal zenginlik ve güçlerine göre bir spektrumda sınıflandırılabilmektedir [21, 22]. Örneğin Taksonomiler ve Eş

Anlamlılar Sözlüğü türündeki ontolojilerde temsil gücü sınırlıyken RDF, DAML+OIL ve OWL türündeki ontoloji dilleriyle tasarlanan bilgi alanlarında kavramlar arasında daha güçlü ilişkiler temsil edilebilmektedir [21]. Tasarım ve modelleme dilleri açısından ontolojilerin metin, grafik vb. teknikler kullanılarak gerçekleştirilmelerinde anlamsal açıdan farklılık bulunmamaktadır. Ayrıca ontoloji dilleri, sözdizim kuralı (syntax) ve anlamsal yapı içeren biçimsel mantığa dayandırılmaktadırlar. Böylece, ontoloji biçiminde temsil edilen bilgi alanları hakkındaki anlamsal yapı ve bilginin temsil gücü artarken buna paralel olarak çıkarsama işlemleri için gerek duyulan işlem, süre ve yapının karmaşıklığı da artmaktadır.

Öte yandan Bilgi Mühendisliği alanındaki ontolojilerin, bilgi yönetimi uygulamalarında, yeni bilgi çıkarımında, veritabanı tasarımı ve entegrasyonu ile akıllı bilgi sistemlerinde yaygın olarak kullanıldığı gözlenmektedir. Çeşitli araştırma ve uygulama alanlarında ontolojik yöntem ve araçlar; (a) farklı sistemler arasındaki iletişim ve birlikte çalışabilirliği sağlamak, (b) bir sistemin yapısıyla ilgili hesaplamaya dayalı çıkarımlarda bulunmak ve (c) süreç boyunca bilginin organizasyonunu, tekrar kullanılabilirliğini sağlamak amacıyla tasarlanmaktadır [17]. Ontolojiler aynı zamanda teorik temel oluşturmak amacıyla çeşitli bilgi alanlarına ait bilgi gösterimlerini değerlendirmek için de kullanılabilmektedir. Bilginin gösterimi ve temsil edilmesi kapsamındaki ontolojiler genellikle şu amaçlar için tercih edilebilmektedir [20]:

- Bilginin tekrar kullanımına yönelik içerik ontolojileri: Genel veya ortak kullanılabilirler, görevlere ve belirli alana yönelik taksonomi ve ontolojileri içerirler.
- Bilgiyi paylaşmak amacıyla (sorgu-cevap) sistemler arasındaki iletişime yönelik ontoloji olarak tasarlanabilirler.
- Çeşitli olayları ve vakaları indekslemeye yönelik ontolojiler olabilir.
- Bilginin gösterimine yönelik üst veri (meta) ontolojilerini oluşturabilirler.

3. İlgili Çalışmalar

Literatür incelendiğinde, YYY'a yönelik çalışmaları genellikle yazılım endüstrisindeki uygulamalar ile deneyim ve teorik çalışmaların oluşturduğu, deneysel araştırmaların ise sınırlı sayıda olduğu gözlenmektedir [22-24]. 1990'lı yılların ikinci yarısından itibaren çeşitli kurumsal yazılımların zaman içerisinde "legacy" sistemlere dönüşmesiyle birlikte YYY'nın önemi artmış, buna yönelik yöntem ve teknikleri içeren çalışmalar ortaya konulmaya başlanmıştır [26-28]. YYY süreçleri, tersine mühendislik, yeniden düzenleme, yazılım bakımı ve yöntemleri ile yazılım iyileştirme araç ve teknikleri belli başlı incelenen konular arasındadır. Bu zaman aralığında çalışılan bir diğer konu da yapısal dillerle geliştirilen mevcut yazılım sistemlerin Nesneye Yönelimli Programlama (NYP) dilleriyle yeniden geliştirilmesidir [29-32]. Mevcut sistemlerin yine YYY çerçevesinde yazılım kalitelerinin iyileştirilmesi, buna yönelik yöntem ile araçların geliştirilmesi ve önerilmesi araştırılan diğer konular arasındadır [2, 3, 14].

Öte yandan, bilgi teknolojileri, internet, veri yönetimi, verinin temsili ve transferiyle ilgili teknoloji ve yaklaşımların gelişmesiyle birlikte yazılım mühendisliğinde çeşitli amaçlar için ontolojilerin de kullanıldığı gözlenmektedir. Ontolojilerin yazılım mühendisliğinde kullanım biçimleri ve yaklaşımları (a) ilgili ontolojinin yazılımın çalışma zamanı (run time) aşamasında kullanılma durumu ile (b) ontolojinin yazılım geliştirme sürecinde

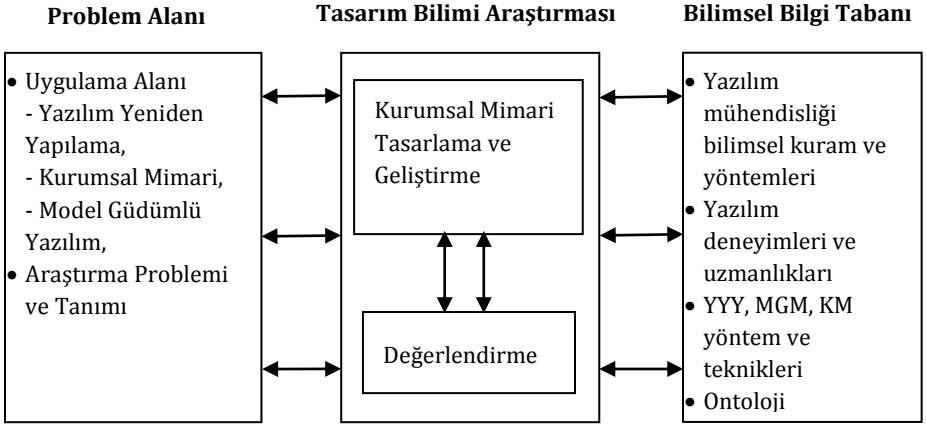
sağladığı desteğe ve bilgiye göre sınıflamak mümkündür [33]. Buna göre:

(1) Ontoloji Güdümlü Yazılım Geliştirme: Bu yaklaşım yazılım geliştirme sürecinin bütününü içermektedir. Ontolojiler, yazılım ihtiyaçlarının tanımlanmasından analiz, tasarım ve yazılımın geliştirilmesine kadar olan süreçte belirleyici bir rol üstlenmektedir.
(2) Ontoloji Destekli Yazılım Geliştirme: Ontolojiler bu yaklaşımda, yazılımın geliştirilmesi süresince yazılım mühendislerine bileşen aramada, problem çözmede vb. görevlerin yerine getirilmesinde sınırlı düzeyde yardımcı olmaktadır.
(3) Ontoloji Tabanlı Yazılım Mimarileri: Çalışma zamanı dikkate alındığında ontolojiler bu yaklaşımda uygulama mantığı açısından yazılımın merkezinde yer almakta, iş kuralları vb. bileşenlerin modellenmesinde kullanılmaktadır.
(4) Ontoloji Destekli Yazılım Mimarileri: Ontolojilerin temel işlevi yazılım alt yapısının desteklenmesidir. Anlamsal web ile mevcut web hizmetlerin desteklemesi, otomatik bilgi arama, hizmet desteği ve yeni yeni yazılım işlevlerinin eklemesi ontolojilerin bu tür kullanımına örnek gösterilebilir.

Sonuç olarak, YYY ve ontolojinin birlikte kullanıldığı literatür incelendiğinde ilgili çalışmaların sınırlı olduğu [34-36], KM çerçevesinde YYY'ı ele alan araştırma veya deneyim çalışmalarının bulunmadığı gözlenmiştir. Daha önce belirtildiği gibi yazılım süreçleri ve altyapı boyutlarında bütüncül yaklaşımları gerektiren bu durum, araştırmamızın temel problemini teşkil etmektedir.

4. Yöntem

Tasarım Bilimi Araştırma Yönteminde (TBAY) (Design Science Research), mühendislik, bilişim sistemleri ve yazılım alanındaki problem alanlarına yönelik, belirli işlev ve özelliklere sahip araç, sistem ve modeller geliştirilirken aynı zamanda bu sistemlerin analizi, tasarımı, geliştirilmesi ve değerlendirmesiyle ilgili bilimsel bilgi birikimi oluşturulmaktadır (Şekil 3) [7]. Bilişim ve bilgisayar



Şekil 3 TBAY Temel Bileşenleri

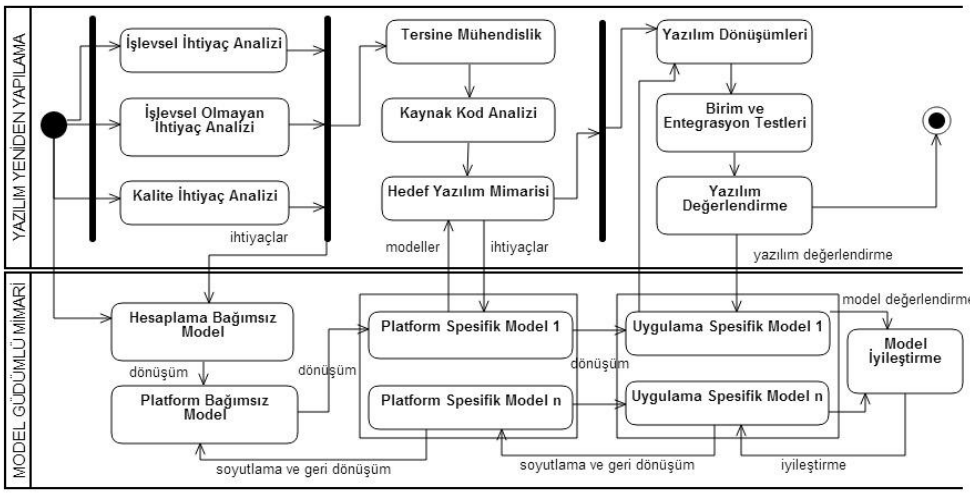
endüstrisinde tasarım ve geliştirme projelerinin temel amacı, mevcut ve onaylanmış standartları, bilgi birikimini rutin süreç ve modelleri kullanarak teknolojik ürünleri maliyet etkin biçimde geliştirmektir. TBAY'de ise bunlara ek olarak bu ürünlerin daha iyi geliştirilmesini sağlayacak bilimsel bilgi birikimine katkıda bulunmak hedeflenmektedir. Dolayısıyla TBAY

dayalı bir araştırma projesinde, gerçek hayat problemlerinden hareket edilerek araştırma yapar gibi bilişim araç, yöntem, model veya kuramları geliştirilir, iyileştirilir ya da sınanır [7, 8]. Şekil 3'te gösterildiği gibi TBAY'de gidiş-dönüşlü ve yinelemeli etkinliklerin bulunduğu; (a) Problem Alanı, (b) Tasarım Bilimi Araştırması ve (c) Bilimsel Bilgi Tabanından oluşan üç ana bileşen bulunmaktadır.

Bu araştırma TBAY çerçevesinde gösterilen modelde YYY ve MGM olmak üzere birbiriyle bütünleşik iki yazılım geliştirme süreç alanı bulunmaktadır. Modelleme ve YYY etkinlikleri bu iki alan arasında gidiş-dönüşlü olmakta, yazılım modülleri ve modeller farklı soyutlama düzeylerinde detaylandırılarak model dönüşümleri gerçekleştirilmektedir.

4.1. YYY'ya Yönelik Kurumsal Mimari ve Geliştirilmesi

KM geliştirme sürecinde TOGAF v.9.1 (The Open Group Architecture Framework) standardı kapsamındaki ADM (Architecture Development Method) mimari geliştirme yöntemi temel alınmış, bu süreç aynı zamanda TBAY'nin tasarım-geliştirme-değerlendirme adımlarıyla da bütünleştirilmiştir. Çalışma süresince ArchiMate, KM geliştirme ortamı ve modelleme dili olarak kullanılmıştır [12]. YYY yönelik geliştirilen KM modeli bütün katmanları içerecek biçimde Ek-A'da gösterilmiş olup YYY KM katmanları ve geliştirme aşamaları aşağıda açıklanmıştır.



Şekil 4. YYY Süreç Modeli [14]

yürütülmüş, YYY süreçlerine yönelik geliştirilen KM araştırmanın çıktısını oluşturmuştur. Çalışmanın TBAY doğrultusunda teorik temelleri ve ana bileşenleri: YYY, MGM, TOGAF v.9.1 Kurumsal Mimari Yaklaşımı ve Ontoloji bilgi alanları ve standartlarıdır. Yazılım mühendisliği problem alanıyla olan ilişkiyi, endüstrideki YYY uygulamaları ile literatür taraması sonucunda ortaya konulan bulgular belirlemiştir. Çalışmamızda KM geliştirme süreçlerine temel olan YYY modeli (Şekil 4) bir önceki araştırmamızda geliştirilmiştir [14]. Buna göre geleneksel YYY faaliyetleri, model güdümlü olarak yürütülmekte, kalite gereksinimleri yine MGM doğrultusunda gerçekleştirilmektedir [15, 16]. Şekil 4'te

4.1.1. YYY Kurumsal Mimari Proje Süreçleri ve MGYG Katmanı

Şekil 5'de gösterildiği gibi YYY sürecinin bu aşaması, ihtiyaç analizinin yapıldığı,

MGM kapsamında hesaplama ve platform-bağımsız modellerin oluşturulduğu, yazılım etkinliklerin bütünleştirilerek paralel yürütüldüğü aşamadır. En genel anlamda kaynak ve hedef yazılımlarının modellenmesi ve hedef yazılıma dönüştürülme süreci aşağıdaki gibi ifade edilebilir:

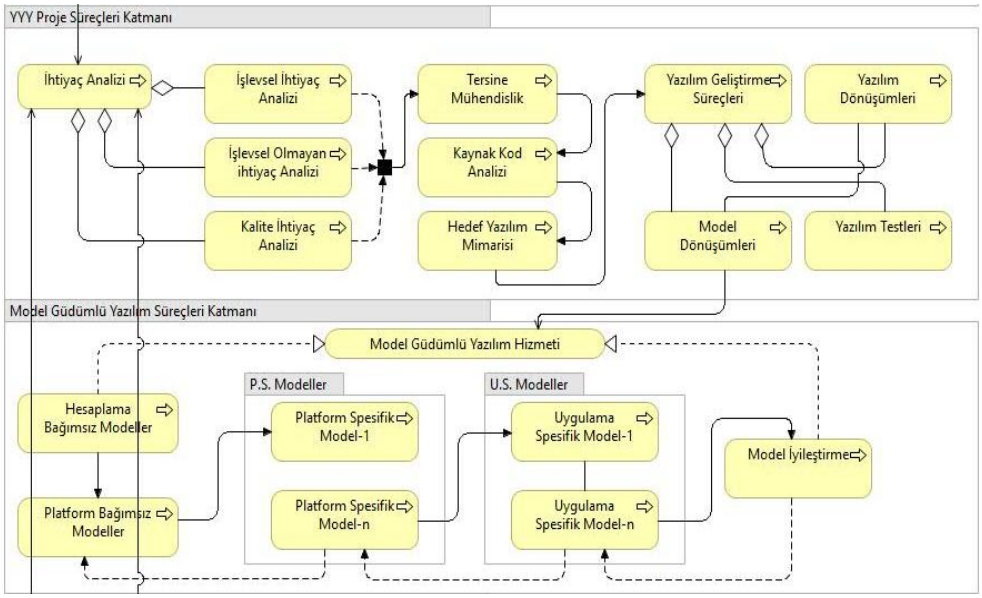
$$t : M_1(S_1) \mid_{F1} \rightarrow M_2(S_2) \mid_{F2} \quad (1)$$

t, YYY sürecindeki t_1, $t_2 \ldots t_n$ sıralı model dönüşümlerini, S_1 yeniden yapılandırılacak kaynak yazılımı, S_2 hedef yazılımı, M_1 kaynak yazılıma ait modelleri, M_2 hedef yazılıma ait modelleri, F_1 ile F_2 ise kaynak ve hedef yazılımın model dönüşümlerinde kullanılan formal gösterim yöntemlerini simgelemektedir.

Hesaplama Bağımsız Modeller olarak oluşturulmaktadır. MGM çerçevesinde Soyut Söz Dizim Ağaçları (SSDA) (Abstract Syntax Tree), mevcut yazılımın kaynak kodlarından platform bağımsız modeller olarak çıkarılmaktadır. Daha sonra bu modeller, modelden modele dönüşüm kuralları, işlevsel ve kalite gereksinimleri de dikkate alınmak suretiyle hedef yazılımda aynı soyutlama düzeyinde SSDA'ları olarak gösterilmektedir. Böylece, mevcut yazılıma ait bütün gereksinimler ile anlam bilimsel yapının hedef yazılıma olduğu gibi aktarılabilmesi aşağıda gösterildiği gibi sağlanmaktadır:

$$M_s \rightarrow M_s{}^a \rightarrow M_t{}^a \rightarrow M_t \quad (2)$$

M_s mevcut yazılıma ait modelleri, $M_s{}^a$



Şekil 5. YYY Proje ve MGYG Süreçleri Katmanı

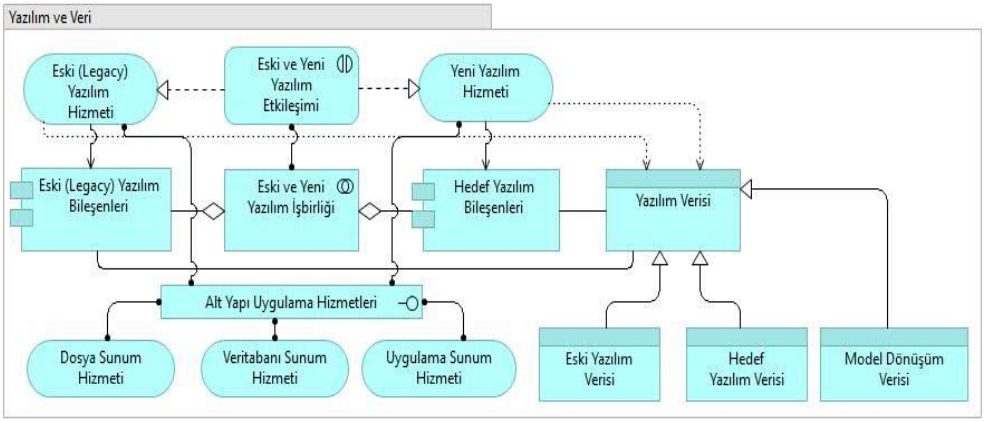
Öncelikle ihtiyaç analizine, işlevsel olan ve olmayan gereksinimlerin belirtimi ile başlanılmaktadır. Doküman incelemesi, tersine mühendislik ve kaynak kod analizi ile iyileştirilecek yazılımın incelemesi yapılır. Mevcut yazılıma ait varsa doküman incelenmesi ya da tersine mühendislik ile iş akış çizenekleri yine mevcut yazılımın SSDA'ları olarak dönüştürülmüş platform bağımsız modelleri, $M_t{}^a$ ise gerekli düzenleme, kalite ve eklemeler yapılıp aynı soyutlama düzeyinde gerçekleştirilen hedef yazılımın SSDA'larını, son olarak M_t ise hedef yazılıma ait dönüştürülmüş modelleri simgelemektedir.

UML çizenekleri de kaynak ve hedef yazılımda platform bağımsız modelleri temsil etmektedirler. Bu çizeneklerde mevcut sistemle ilgili yazılım mimarisi, yapısal ve işlevsel hatalar belirlenerek düzeltilmektedir. Yazılım kalite standartları çerçevesinde uygun metrik ve ölçütlerle hedef yazılımın kalite gereksinimleri ortaya konulmaktadır. Böylece, hedef yazılımın sahip olması istendiği P_1, P_2, …P_n özellikleri ve $KÖ_1$, $KÖ_2$, …$KÖ_n$ kalite ölçütleri MGM çerçevesinde platform bağımsız olarak modellenmektedir.

4.1.2. YYY Kurumsal Mimari Yazılım ve Veri Katmanı

Bu aşama yazılım bileşenlerinin birbirleriyle ilişkilendirildiği ve geliştirildiği safhadır (Şekil 6). İhtiyaç analizinde belirlenmiş işlevsel/işlevsel olmayan gereksinimler ve yazılım kalite ihtiyaçları hedef programlama dili de hizmetleri arasındaki ilişkileri gösteren KM katmanı verilmiştir.

Öncelikle hedef programlama dili ve yazılım çalıştırma platformu dikkate alınır, bir önceki aşamadaki platform-bağımsız model olarak oluşturulan SSDA'ları, UML çizenekleri ve Nesneye Yönelimli Programlama (NYP) metrikleri, yazılım kalite ölçütleri doğrultusunda hedef yazılım mimarisi belirlenmektedir. Bu aşamada aynı zamanda SSDA'ları ile anlamsal bütünlük, NYP metrikleri ile yapısal bütünlük ve yazılım kalite ölçütleri karşılanmaktadır. Aslında bunların MGYG'deki karşılığı düzenleme dönüşümleri (refactoring transformation) ya da modelden modele (model-to-model transformation) dönüşümlerdir. Model dönüşüm yöntemleri ve yazılım desenleri (pattern) bu dönüşümlerde yol gösterici ilkeleri ortaya koyarak yazılım geliştirme ve



Şekil 6. YYY Kurumsal Mimarisi Yazılım ve Veri Katmanı

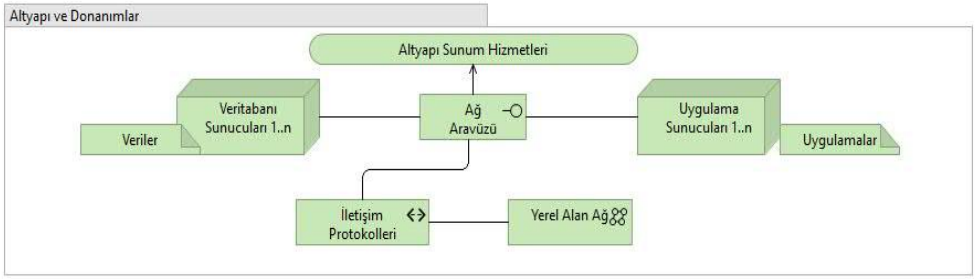
dikkate alınarak hedef yazılım mimarisi ve model dönüşüm kurallarına göre belirlenir. Başka bir ifadeyle, farklı seviyelerdeki MD_1, MD_2, …MD_n model dönüşümlerini içerecek YD_1 YD_2, …YD_n yazılım dönüşümleri yinelemeli ve artırımsal olarak gerçekleştirilmektedir. Şekil 6'da eski ve yeni yazılım bileşenleri, bunlara ait veriler, bileşen ve yazılım iyileştirme sürecine de formalizm kazandırmaktadır. Daha sonra hedef programlama dili ve yazılım mimarisinden hareket edilerek iyileştirilmiş platform-bağımsız modeller (UML), platform-spesifik modellere dönüştürülürler. Diğer bir ifadeyle geliştirilen modeller hedef programla diliyle (Java, C#, XML vb.) gösterilmektedir. Bu yazılım

dönüşümleri, elle veya otomatik, UML profili ve yazılım desenleri kullanılarak gerçekleştirilmektedir.

Mevcut ve iyileştirilen yazılıma ait platform-spesifik modeller, hedef platforma yönelik daha da detaylandırılmakta, MGM çerçevesinde uygulama-spesifik model olarak ifade edilen bilgisayar kodlarına dönüştürülmektedir (model-to-code transformation). Birim ve entegrasyon testleriyle yeniden yapılanan yazılım sistemi S', geçerleme ve doğrulama süreçlerinden geçirilmekte, P_1,(S'), P_2,(S'), …, Pn,(S') yazılım özellikleri ile $KÖ_1$,(S'), $KÖ_2$,(S'), …, KÖn,(S') kalite ölçütlerinin ne kadar karşıladığı bu aşamada değerlendirilmektedir. Değerlendirme sürecinde yapılan güncellemeler ve düzenlemeler, uygulama ve platform-spesifik modellere tekrar aktarılmaktadır. Böylece model geri dönüşümleri ve soyutlama kullanılarak platform ve hesaplama bağımsız modellere söz konusu değişiklikler yansıtılmaktadır.

4.1.3. YYY Kurumsal Mimari Altyapı ve Donanım Katmanı

Bu katman, yeniden yapılanacak hedef yazılımın, mevcut ve yeni teknolojilerle birlikte bütünleşik olarak bulunduğu ve işletildiği katmandır (Şekil 7).



Şekil 7. YYY Kurumsal Mimari Altyapı ve Donanım Katmanı

Endüstrideki uygulamalar ve literatürdeki çalışmalar incelendiğinde çoğunlukla YYY'da yazılım süreçleri ve kullanılan araçlar üzerinde durulduğu gözlenmektedir. Araştırmalar, "legacy" yazılım sistemlerinin işlevsel ve kalite ihtiyaçlarına ek olarak, donanım ve altyapı teknolojilerinde de belirli ölçülerde değişiklik ve güncellemelere ihtiyaç duyulabildiğini göstermektedir. Üstelik bu altyapı ihtiyaçları, ilgili kurumun diğer BT ihtiyaçlarıyla da birlikte ele alınabilmektedir. Bazen bir YYY projesi, veritabanı göçü ve sistem iyileştirme gibi ek projelerle çok daha kapsamlı ve karmaşık bir BT projesine evirilebilmektedir. Dolayısıyla bu durumun, YYY sürecinin KM çerçevesinde alt yapı, donanım ve diğer teknolojik ihtiyaçlarla da birlikte ele alınması gerektirdiğini söylemek mümkündür.

4.2. YYY Yönelik Kurumsal Mimarinin Test ve Değerlendirilmesi

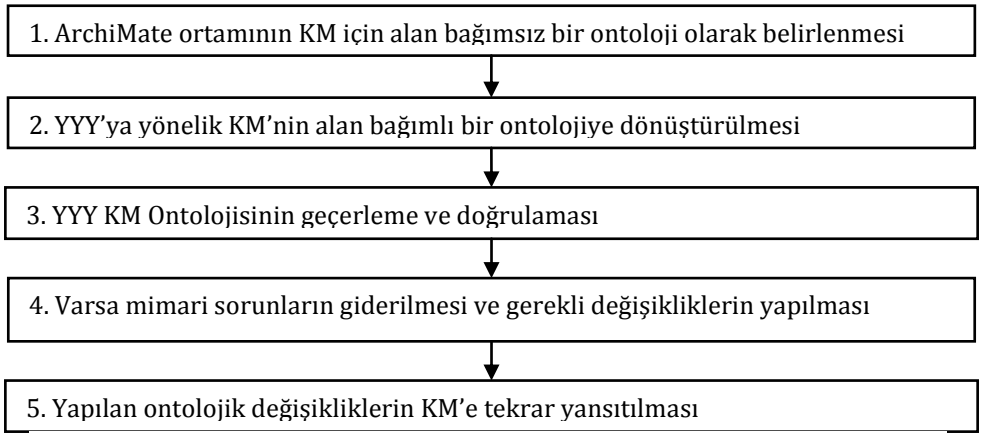
Makalenin bu bölümüne kadar olan kısmında, araştırma problemi doğrultusunda geliştirilen YYY model güdümlü süreç modeli ile KM'nin bileşenleri tanıtılmıştır. Ancak, ArchiMate geliştirme ortamı ve modelleme diliyle geliştirilen bu KM'nin birkaç yönüyle de değerlendirilmesi gerekmektedir. Birincisi, önerilen KM'de yer alan YYY yapıları ve bileşenlerinin bilgiyi temsil gücü nedir, geliştirilen bu modeldeki bileşenler anlamsal olarak ne ölçüde birbirlerine bağımlı ve kendi içerisinde tutarlıdırlar? İkincisi, geliştirilen bu KM model, kendi mimari bileşenleri olan iş süreçleri, uygulama ve altyapı katmanları arasında iletişime,

bilgi alışverişine ve birlikte çalışabilirliğe ne ölçüde olanak tanımaktadır?

Daha önce belirtildiği gibi taksonomilerde bir bilgi alanı, en genelden en özele doğru hiyerarşik biçimde sadece altsınıf ve üstsınıf ilişkilerle temsil edilebilmektedir. Oysa ontolojiler taksonomiler üzerine yapılandırılarak bilginin anlamsal olarak daha iyi temsil edilebilmesine, yeni ilişki ve özelliklerin kullanılabilmesine olanak vermektedir [20]. Bu bağlamda, çalışmamızda önerilen YYY modeli ile KM'deki farklı yapıdaki bileşen ve katmanlar arasında göreceli olarak karmaşık ilişkiler bulunmakta, aynı zamanda yoğun bir bilgi alışverişine de ihtiyaç duyulmaktadır. Dolayısıyla, söz konusu KM'nin değerlendirilmesi sürecinde ontolojik analiz ve değerlendirme yöntemlerinin belirtilen ihtiyaçlara cevap verebileceği gözlenmiştir.

Literatürde çeşitli üst seviye bilgi gösterim ontolojileri ve ontoloji tasarım dilleri mevcut olup Çerçeve (Frame), OKBC, RDF, OIL, DAML+OIL vb. diller bunlara örnek gösterilebilir. Bunlardan ayrı olan OWL (Web Ontology Language), World Wide Web Consortium (W3C) tarafından Anlamsal Web çerçevesinde DAML+OIL'den türetilerek RDF üzerine yapılandırılmış ve bir standart olması hedeflenmiştir [20]. Ayrıca ontoloji alanındaki araştırmalar ile endüstrideki uygulamalar dikkate alındığında OWL'nin yoğun olarak kullanıldığı gözlenmektedir. Dolayısıyla, bu çalışmada önerilen KM'nin değerlendirilmesi sürecinde OWL'nin kullanılmasının uygun olacağı düşünülmüştür. YYY alanına yönelik geliştirilen KM'nin anlamsal bütünlüğü, tutarlılığı ve bilgiyi temsil etme gücü ontolojik yöntemlerle Şekil 8'de belirtilen adımlar izlenerek test ve değerlendirilmiştir [18].

4.2.1. Archimate Ortamının KM İçin Alan Bağımsız Bir Ontoloji Olarak Belirlenmesi



Sekil 8. YYY Kurumsal Mimarisinin Ontolojik Olarak Değerlendirilme Süreci

Öncelikle birinci adımda ArchiMate KM geliştirme ortamı alan bağımsız bir ontoloji olarak belirlenmiştir. Bu ortamda YYY yönelik KM geliştirildikten sonra alan bağımlı bir ontolojiye dönüştürmek için özel bir eklenti kullanılmıştır [18]. Her ne kadar söz konusu eklentinin belirli ölçüde sınırlılıkları olsa da KM'nin çekirdek bileşenler ve aralarındaki ilişkileri

ontolojik yapılar biçiminde temsil edilebildiği gözlenmiştir.

4.2.2. YYY'ya Yönelik KM'nin Alan Bağımlı Bir Ontolojiye Dönüştürülmesi

İkinci adımda söz konusu KM, Anlamsal Web Dili (Web Ontology Language/OWL) kullanılarak YYY KM'ni temsil eden bir alan bağımlı ontolojiye dönüştürülmüştür. Yer sınırlamasından dolayı RDF tanımlamalarının sadece başlangıç bölümü Şekil 9'da gösterilmiştir.

Protégé ve Archimate) meta modelleri OWL dilinde ontolojiye çevrilebilmekte, böylece eşleştirme, anlamsal sorgulama ve tutarlılık kontrolleri otomatik hale getirilebilmektedir. Dolayısıyla bu aşamada geliştirilen KM ontolojisi Protégé ortamına aktarılmış (Ek-B), SPARQL (Simple Protocol for RDF Query Langauge) eklentisi ve sorgulamaları ile (a) tutarlılık, (b) sınıflama ve (c) anlamsal bütünlük açısından geçerleme ve doğrulamaları yapılabilmiştir. Tablo 1'de YYY'a yönelik KM'e ait ontolojinin yapısı ve kendisini oluşturan bileşenler

```
<rdf:RDF
# ********* **YYY Kurumsal Mimarisinin tanımlandığı RDF dökümanı 28.05.2016**********
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:j.0="http://pszwed.kis.agh.edu.pl/ontologies/archi/archimate.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description
rdf:about="http://pszwed.kis.agh.edu.pl/ontologies/archi/YYY_Mimarisi.owl#809268df">
   <j.0:flow
rdf:resource="http://pszwed.kis.agh.edu.pl/ontologies/archi/YYY_Mimarisi.owl#9f4a6e44"/>
   <j.0:triggering
rdf:resource="http://pszwed.kis.agh.edu.pl/ontologies/archi/YYY_Mimarisi.owl#9a9e3164"/>
   <rdfs:label>Platform Spesifik Model-n</rdfs:label>
   <rdf:type
rdf:resource="http://pszwed.kis.agh.edu.pl/ontologies/archi/archimate.owl#BusinessProcess"/>
  </rdf:Description>
# * RDF dokümanının bundan sonraki bölümleri makaleye konulmamıştır. *
```

Sekil 9. KM Ontolojisinin Tanımlandığı RDF Dökümanı

4.2.3. YYY KM Ontolojisinin Geçerleme ve Doğrulaması

Ontolojilerin önemli özelliklerinden birisi de genel amaçlı çıkarsama motorları kullanılarak modellenen alanla ilgili mantıksal çıkarsamalara ve sorgulamalara olanak tanımalarıdır. [36]'da görüleceği gibi farklı geliştirme ortamlarının kendilerine ait (Örneğin, arasındaki ilişkileri sorgulayan iki anlamsal sorgu örnek olarak verilmiştir.

Tablo 1'deki 1 numaralı sorguda, geliştirilen KM'de hangi ilişki türlerinin bulunduğu araştırılmaktadır. Sorgulama sonucunda; "realization", aggregation", "triggering, "access", "association", "assignment", "specialization", "used by"

ve "flow" ilişki türlerinin olduğu belirlenmiş olup "composition", "grouping", "junction" ve influence" ilişkilerinin olmadığı görülmüştür. Tabloda verilen şekillerde yer sınırlaması dolayısıyla sadece ilk beş ilişki gösterilebilmiştir. Aslında, Archimate ile tasarlanan KM'deki ilişki türlerinin neler olduğu ile nasıl belirlendiği bu çalışmanın kapsamı dışındadır. Ancak, bu çalışmada önerilen KM'de bulunmayan ise YYY kapsamındaki ihtiyaçlara bağlı olarak iş süreçleri, yazılım ve donanım katmanlarındaki bileşenlerin biri veya birkaçında, birbirlerini doğrudan etkilemeyecek ekleme, çıkarma, yazılım güncellemesi veya değişikliğinin yapılabileceği biçiminde yorumlanabilir". Öte yandan, Tablo 1'deki 2 numaralı sorgu daha özel amaçlı olup YYY'a yönelik KM'deki bileşenler arasındaki "alt sınıf-üst sınıf" veya "baba-oğul"

Tablo 1. KM'e Ait Ontolojinin Yapısıyla İlgili Örnek SPARQL Sorguları

	Anlamsal Sorgu ve Amacı	**Sorgulama Sonucu**
1	*Amaç*: Geliştirilen KM'deki bileşenler arasındaki ilişki türleri ile KM'nin hangi ortamda geliştirildiğini bulmak. ---------- *Sorgu*: SELECT ?iliskiTurleri ?GelistirmeOrtami WHERE {?iliskiTurleri rdfs:subPropertyOf ?GelistirmeOrtami}	SPARQL query: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> SELECT ?iliskiTurleri ?GelistirmeOrtami WHERE {?iliskiTurleri rdfs:subPropertyOf ?GelistirmeOrtami} iliskiTurleri realisation archimate aggregation archimate triggering archimate access archimate association archimate Execute
2	*Amaç*: Geliştirilen KM'deki alt sınıf-üstsınıf ilişkisi bulunan bileşenleri bulmak. ---------- *Sorgu*: SELECT ?cocukSinif ?babaSınıf WHERE {?cocukSinif rdfs:subClassOf ?babaSınıf	SPARQL query: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> SELECT ?cocukSinif ?babaSınıf WHERE { ?cocukSinif rdfs:subClassOf ?babaSınıf } cocukSinif babaSinif ApplicationService ApplicationLayerElement Artifact TechnologyLayerElement ApplicationInteraction ApplicationLayerElement InfrastructureInterface TechnologyLayerElement BusinessRole BusinessLayerElement Execute

"composition" ilişkisi ile ilgili dolaylı olarak şöyle bir yorum çıkarılabilir: "Ek 1'de gösterilen KM'de, gerek bileşenler ve elemanlar arasında, gerekse süreç, uygulama ve teknoloji/altyapı katmanları arasında ilişki türlerinin en güçlüsü olan "composition" ilişkisi yoktur. Bir başka ifadeyle, var olması birisinin diğerine doğrudan bağlı, "alt sınıf-üst sınıf" ilişki türündeki herhangi bir süreç, yazılım ve donanım bileşeni bulunmamaktadır. Bu ilişkisi bulunan bileşenleri temsil eden sınıflar belirlenmektedir.

4.2.4. Mimari Sorunların Giderilmesi ve Değişikliklerin KM'ye Yansıtılması

Bu aşamada, ontolojik analiz sonucunda belirlenen tasarım hataları veya yeni gereksinimler yine Protégé ortamında yapılan düzenlenmelerle KM ontolojisine aktarılmaktadır. Güncellenen KM model

ve ontoloji, MGYG çerçevesinde tersine dönüşüm işlemleriyle; (a) ArchiMate ortamında alan bağımsız modellere, (b) UML çizenekleri kapsamında platform bağımsız modellere dönüştürülmektedir.

Sonuç olarak, ontolojik değerlendirme sonuçlarından hareket edilerek; (a) önerilen YYY'ya yönelik KM'nin ilgili süreçlerini belirli ölçüde soyutlayabildiğini; (b) YYY ile ilgili ontolojinin yapısal ve anlamsal temellerinin olduğunu; (c) bu ontolojinin hesaplamalı çıkarım ve karar verme süreçlerinde kullanılabildiğini söylemek mümkündür.

4.3. Çalışmanın Sınırlılıkları

TBAY'nin önemli bir bileşeni test ve değerlendirme sürecidir. Çalışmanın sınırlılıkları çerçevesinde bu araştırmada geliştirilen KM'nin, örnek olay, durum çalışması vb. deneysel yöntemlerle sınanması mümkün olmamıştır. Bu çalışmada önerilen KM'in anlamsal yapısı ve bilgiyi temsil gücü ontolojik yöntemlerle analiz edilmiş, bilgisayarla işlenebilir hale getirilerek değerlendirilmiştir. Dolayısıyla, çalışma sonuçlarının genellenebilirliği bu yönüyle sınırlı düzeydedir. Araştırmanın iç geçerliliğini tehdit edebilecek unsurlar uzman görüşlerine başvurularak ve literatüredeki çalışmalar incelenerek giderilmeye çalışılmıştır. Bir diğer sınırlılık ise geliştirilen modelin nesneye yönelimli yazılım sistemleri için olması ve yapısal programlamayla geliştirilmiş sistemlerin ihtiyaçlarına cevap verebilecek nitelikte olmamasıdır.

5. Sonuç ve Öneriler

Bu makalede, araştırmanın genel amacı ve problemi doğrultusunda YYY ile ilgili olarak üç alt problem ortaya konulmuştur. Bunlar sırasıyla:

- Yoğun iş gücü ve kaynak gerektiren YYY süreçlerinin otomatik hale getirilebilmesi, üretilen bileşen ve çıktıların sonraki aşama ve süreçlerde kullanılabilmesinin sağlanması,

- Bir YYY projesinde sadece yazılıma yönelik değil, aynı zamanda ilgili kurumda değişen süreç, veri ve teknolojiyle ilgili yeni gereksinimlerin de dikkate alınması zorunluluğu,

- YYY, MGYG ve KM araştırma alanı çerçevesinde; (a) süreçler arasında bilgi paylaşımına, (b) farklı platformlardaki bileşenlerin entegrasyonuna ve (c) bunlar arasındaki yapısal ve kavramsal boyuttaki ilişkilerin ortaya konulmasına olanak tanıyacak anlamsal yöntem ve araçlara olan ihtiyaçlardır.

Bu amaçla, söz konusu problem sahalarına yönelik TBAY çerçevesinde bir araştırma yürütülmüş ve YYY süreçleri için bir KM geliştirilmiştir. KM'nin teorik temelleri ve ana bileşenlerini YYY, MGM, TOGAF KM ve Ontoloji bilgi alanlarındaki yöntem, teknik ve araçlar oluşturmuştur. İlk izlenimlerimiz, geliştirilen bu KM'nin değişik soyutlama düzeylerindeki YYY problemlerine farklı bakış açılarını kazandırdığı ve YYY ile ilgili paydaşların görüş ve ihtiyaçlarına cevap verebileceği yönündedir. Ancak, araştırmanın sınırlılıklarında belirtildiği gibi geliştirilen KM'nin deneysel yöntemlerle sınanması mümkün olamamıştır. Dolayısıyla makalemiz, bu KM'nin uygulandığı, test ve değerlendirilerek sonuçların daha ayrıntılı biçimde tartışıldığı endüstri uygulamaları ile deneysel yazılım çalışmaların yapılması önerisiyle son bulmaktadır.

Kaynakça

[1] Editorial, 2011. A retrospective view of software maintenance and reengineering research- a selection of papers from 2010 European

Conference on Software Maintenance and Reengineering. *Journal of Software Maintenance and Evolution*, DOI: 10.1002/smr.548.

[2] Tahvildari, L., Kontogiannis, K. & Mylopoulos J. 2003. Quality-driven software reengineering, *The Journal of Systems and Software*, 66, s.225-239.

[3] Uysal, M.P. ve Mergen, 2013. E. A Quality-oriented approach to software reengineering, *The Northeast Decision Sciences 2013 Annual Conference*, Brooklyn, NY, USA, April 5-7, s.971-979.

[4] Wagner C. 2014. *Model-driven software migration: a methodology, reengineering, recovery and modernization of legacy system*, USA, Springer Vieweg.

[5] Swithinbank, P., Chessell, M., Gardner, T., Griffin, C., Man, J., Wylie, H. & Yusuf, L. 2005. *Patterns: Model-driven development using ıbm rational software architect*, USA, Redbooks.

[6] Beydeda, S.,·Book M., Gruhn, V. 2004.. *Model-driven software development*, USA, Springer-Verlag Berlin Heidelberg.

[7] Hevner, A. & Chatterjee S. 2010. Design Research in information systems, *Integrated Series in Information Systems*, 22, DOI 10.1007/978-1.

[8] Vaishnavi, V.K. & Kuechler W.J. 2008. *Design Science Research methods and patterns: innovating ınformation and communication technology*, USA, Auerbach Publications, Taylor & Francis Group.

[9] Elliot, J. Chikofsky and James H. C. 1990. Reverse engineering and design recovery: a taxonomy, *IEEE Software*, Cilt. 7, No.1, s.13-17.

[10] Object Management Group 2003. *MDA Guide Version 1.0.1. Technical Report* omg/2003-06-01, OMG.

[11] McCall, J. A., Richards, P. K., Walters, G. F. 1977. Factors in software quality, *Nat'l Tech. Information Service*, Cilt. 1, No.2 ve 3.

[12] Lankhorst, M. 2009. *Enterprise architecture at work: modelling, communication, and analysis,* Springer-Verlag Berlin Heidelberg.

[13] TOGAF 2011. *TOGAF Version 9.1, Open Group Standard,* The Open Group.

[14] Uysal, M.P, Mergen E.A. 2015. Yazılım yeniden yapılamaya yönelik model güdümlü ve kaliteye yönelimli süreç modeli, *9. Ulusal Yazılım Mühendisliği Sempozyumu*.

[15] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., Merritt 2016. *M.ISO/IEC 2501n. Quality model division.* http://www.iso.org. Erişim Tarihi: 28.05.2016.

[16] Antunes, G., Bakhshandeh, M., Mayer, Rudolf, Borbinha, J. Caetano,.A. 2013. Using ontologies for enterprise architecture analysis, *17th IEEE International EDOCW*, Vancouver, BC.

[17] Green, P., Rosemann M. 2005. *Business systems analysis with ontologies*, USA, Idea Group Publishing.

[18] Szwed P. 2016. *Plugin for transforming Archimate files into OWL*, [http://home.agh.edu.pl/ ~pszwed /en/doku.php? id=archi_to_owl], Erişim Tarihi: 12.03.2016.

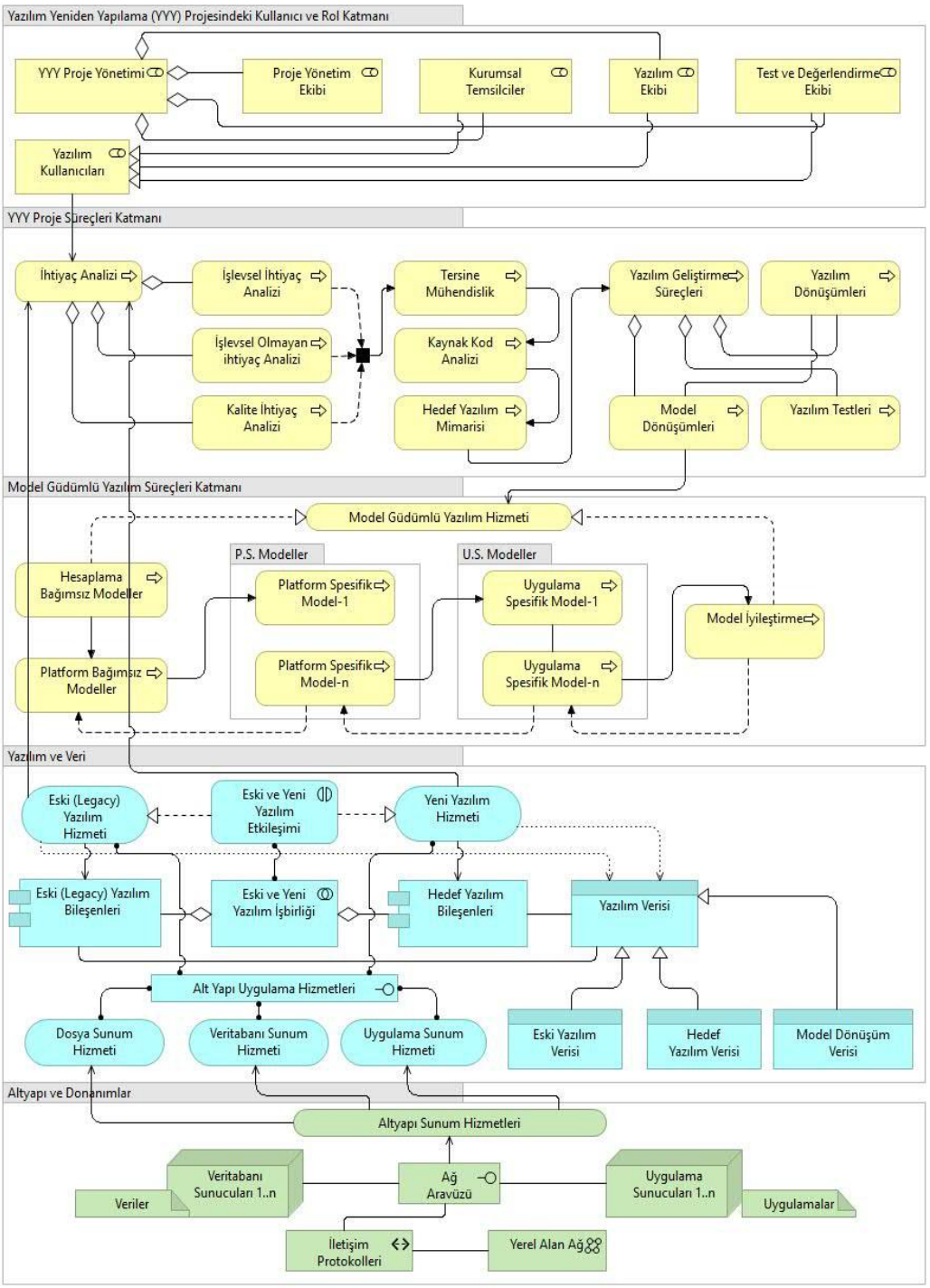
[19] Pérez A.G., López, M.F., Corcho, O. 2004. *Ontological engineering: with examples from the areas of knowledge management, e-Commerce and the semantic web*, Springer-Verlag London Limited.

[20] Daconta, M.C., Smith, K.T., Obrst, L.J., 2003. *The Semantic Web: A guide to the future of XML, web services, and knowledge managemen*t, John Wiley & Sons Inc..

[21] Giray, G. ve Ünalır, M.O. 2007. Yazılım mühendisliğinde

ontolojilerin kullanımı, *III.Ulusal Yazılım Mühendisliği Sempozyumu - UYMS 2007*, Ankara, s.69-76.
[22] Hannay, J.E., Sjøberg, D.I.K., Dybå, T. 2007. A systematic review of theory use in software engineering experiments, *IEEE Transactions on Software Engineering*, Cilt 33(*2*), s. 87-107.
[23] Shull F., Singer, J., Sjøberg, D.I.K. 2008. *Guide to advanced empirical software engineering*, Springer-Verlag London Limited, USA.
[24] Woods, S.G., Quilici, A.E., Yang, Q. 2012. *Constraint-based design recovery for software reengineering: theory and experiments*, Springer, USA,
[25] Miller H. 1997. *Reengineering legacy software systems*, Digital Press, USA,
[26] Sage A.P. 1995. Systems Engineering and systems management for reengineering, *Journal of Systems Software*, Cilt 30, s.3-25,
[27] Birchall C. 2016. *Re-engineering legacy software*, Manning Publications,
[28] Seacord R.C., Plakosh D., Lewis G.A. 2003. *Modernizing legacy systems: software technologies, engineering processes, and business practices*, Addison-Wesley, USA.
[29] Frakes W.B., Kulczycki G., Moodliar N. 2008. *An empirical comparison of methods for reengineering procedural software systems to object-oriented systems*, Springer-Verlag, USA.
[30] Serge D., Ducasse S., Nierstrasz, O. 2002. *Object-oriented reengineering patterns (the morgan kaufmann series in software engineering and programming*), Morgan Kaufmann, USA.
[31] Rada, R. 2005. Reengineering Software: *How to reuse programming to build new, state-of-the-art software*, Glenlake Publishing Co.
[32] Valenti, S. 2002. *Successful software reengineering*, IGI Global, USA.
[33] Happel, H.J., Seedorf, S. 2006. Applications of ontologies in software engineering, *2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006),* Athens, GA, U.S.A..
[34] Yang, H., Cui, Z. ve OBrien, P. 1999. Extracting ontologies from legacy systems for understanding and re-engineering, *Computer Software and Applications Conference*.
[35] Bringuente, A.C.O., Falbo R.A., Guizzardi, G. 2011. Using a foundational ontology for reengineerıng a software process ontology, *Journal of Information and Data Management*, Vol 2, No 3, s.511-526.
[36] Katasonov, A. 2012. Ontology-driven software engineering: beyond model checking and transformations, *International Journal of Semantic Computing*, Vol. 6, No: 2, s.205-242.

Ek-A. YYY'ya Yönelik Kurumsal Mimari



Ek-B. YYY'ya Yönelik Kurumsal Mimarinin Protégé Ortamında Test ve Değerlendirilmesi