Üç Katmanlı Nesne-İlişkisel Eşleme Mimarisi İçin Otomatik Fonksiyonel Büyüklük Ölçümü

Hamdi DEMİREL[1] and Barış ÖZKAN[2]

[1]Atılım Üniversitesi, Yazılım Mühendisliği Bölümü, Ankara, Türkiye
h.demirel@hvkk.tsk.tr
[2] Atılım Üniversitesi, Bilişim Sistemleri Mühendisliği, Ankara, Türkiye
baris.ozkan@atilim.edu.tr

Özet. Fonksiyonel Büyüklük Ölçümü'nde, çeşitli fonksiyonel büyüklük ölçüm metodları kullanılarak objektiflik ve tutarlılığın artırılması için standartlaşmaya ihtiyaç duyulmaktadır. Bazı mimari stiller için kullanımda olan çeşitli standardizasyon tanımları yapılmış olsa da, objektiflik ve tutarlılığı artırmak için hala yapılabilecek iyileştirmeler bulunmaktadır. Bu iyileştirmeler, otomasyon methodu kullanılarak ölçümcü hatalarının, zaman/efor miktarının ve yazılım geliştirme ortamına bağımlılığın azaltılması ile sağlanabilir. Bu çalışmada; Üç-Katmanlı Nesne İlişkisel Eşleme Mimarisi kullanan iş yazılımları için bir yaklaşım oluşturulmuş, prototip bir araç tasarlanmış ve bu araç yardımı ile bulunan sonuçlar, yaklaşımın doğrulanması için deneysel bir çalışma ile sunulmuştur.

Anahtar Kelimeler: COSMIC, Nesne İlişkisel Eşleme, Üç Katmanlı Mimari, Fonksiyonel Büyüklük Ölçümü, Otomasyon.

1 Giriş

Fonksiyonel Büyüklük Ölçümü (FBÖ), yazılım yönetim sürecinde özellikle zaman, efor ve maliyet gereksinimlerinin tahmini için kullanılan temel bir uygulama alanıdır [1, 2, 3]. Fonksiyonel büyüklük, proje yöneticisine proje için kullanılacak olan zaman/efor ve maliyet miktarları, proje takvimlendirme çalışmaları ve üretim parametreleri ile ilgili olarak bir çok değerli bilgi sunmaktadır. FBÖ'de karşılaşılan önemli zorluklardan birisi; FBÖ'nde kullanılan konsept ve kuralların, farklı yazılım mimarileri, programlama dilleri ve kişiden kişiye farklılık gösteren implementasyon detaylarına uyarlanmasıdır [4, 5]. Yazılımın büyüklük ölçümü açısından kabul edilebilir objektiflik ve tutarlılık seviyelerine ulaşılabilmesi için bu problemin çözümü büyük bir önem arzetmektedir. Çeşitli mimari stilleri, mimari paternleri, programlama dillerini ve mimari özellikerli içine alacak şekilde, ölçümlerin standart hale getirilmesi için bazı mimari ölçüm teknikleri ve ölçüm yaklaşımları geliştirilmiştir. Bu teknik ve yaklaşımlardan yaygın olarak kullanılanlardan COSMIC (Common Software Measurement International Consortium) İşlevsel Büyüklük Ölçme Yöntemi iş yazılımları için kullanılan 3-katmanlı mimarilere sık sık başvurmaktadır [6]. Buna ek olarak, COSMIC ayrıca SOA (Service Oriented Architecture) tabanlı uygulamalar için ölçüm metodları da sunmaktadır [7]. Diğer

bir yaklaşımda ise MDA (Model Driven Architecture) tabanlı yazılımların ölçümü için Nesne Yönelimli Metod ve COSMIC FPA (Function Points Analysis) kullanılmıştır [8].

Fonksiyonel Büyüklük Ölçümünde karşılaşılan diğer bir zorluk ise manuel ölçüm yöntemlerinin genellikle fazlaca zaman ve efor gerektirmesi ve ölçümcü hatalarına karşı hassas olmasıdır. Buna karşın otomatik bir ölçüm süreci; zaman, efor ve maliyetin azaltılması ve objektiflik, tekrarlanabilirlik ve tutarlılığın artırılması açısından bir çok avantaj sağlayacaktır [9, 10, 11, 12, 13, 8, 14, 15, 16, 17].

Bu çalışmada; yukarıda belirtilen problemlere yönelik olarak, Üç Katmanlı Nesne-İlişkisel Eşleme Mimarisi kullanan bir iş yazılımı için otomatik bir ölçüm yaklaşımı sunulmuştur. Sunulan yaklaşım için, standart bir tanımının olması (ISO/IEC 19761: 2011 [6]), uluslar-arası geçerliliğinin bulunması ve kullanılan mimari için detaylı bir açıklama sunmuş olması nedeni ile COSMIC FBÖ metodu seçilmiştir.

Bu çalışmadaki motivasyonumuz iki gözleme dayanmaktadır. Bu gözlemler; Nesne-İlişkisel geliştirme ortamının en popüler yazılım mimarilerinden biri olan ve COSMIC tarafından sıklıkla kaullanılan 3-Katmanlı Yazılım mimarisinin, uygulama yazılım bileşenlerinin (veri hareketleri, katman, kapsam ve ayrıştırma seviyeleri vb.) detaylı bir şekilde tespit edilmesi için imkan sunması ve Nesne-İlişkisel Eşleme mimarisinin ölçümcüyü uygulama yazılımı içerisinde kullanılan ve COSMIC ölçümünün yapı taşlarından olan nesnelerin (sınıflar) veri tabanına yazılması ve okunması için gerekli implementasyon detaylarından soyutlamasıdır. Bu yüzden Nesne-İlişkisel Eşleme modeli kullanan iş yazılımları, COSMIC yazılım modeli bağlamında belirtilen yazılım bileşenlerinin sistematik bir analiz ile tespit edilmesi ve İşlevsel Kullanıcı Gereksinimleri'nin (İKG) mimari bileşenler ve implementasyon için kullanılan koddan çıkartılması için fırsatlar sunmakatadır.

Bu çalışmadaki motivasyonumuz ile ilgili olarak sunulan yaklaşımın iki temel amacı:

- 3-Katmanlı Nesne İlişkisel Eşleme mimarisi kullanan İş Yazılımları ile otomatik yazılım ölçümü için çalışmamıza temel oluşturan COSMIC Yazılım Modeli arasında bir ilişki kuramak ve
- Bu ilişkiyi kullanarak otomatik ölçümler gerçekleştiren prototip bir araç geliştirmektir.

Önerilen bu ilişkide, 3-katmanlı mimariler ve nesne ilişkisel konsept için COSMIC tarafından yapılan tanım ve açıklamalar dikkate alınmıştır. Geliştirilen prototip araç için belirli bir mimari ve programlama dili implement edilmiştir.

İkinci bölümde 3-Katmanlı mimari ve Nesne-İlişkisel Eşleme modeli yüzeysel olarak anlatılmıştır. Üçüncü bölümde, COSMIC yazılım modeli ile 3-Katmanlı Nesne-İlişkisel Eşleme arasındaki ilişki detaylandırılmış ve prototip ölçüm araçı tanıtılmıştır. Dördüncü bölümde ise araç tarafından gerçekleştirilen deneysel bir ölçüm yapılmış ve sonuçları ölçümcüler tarafından yapılan manuel ölçüm sonuçları ile karşılaştırılmıştır.

2 Arka Plan

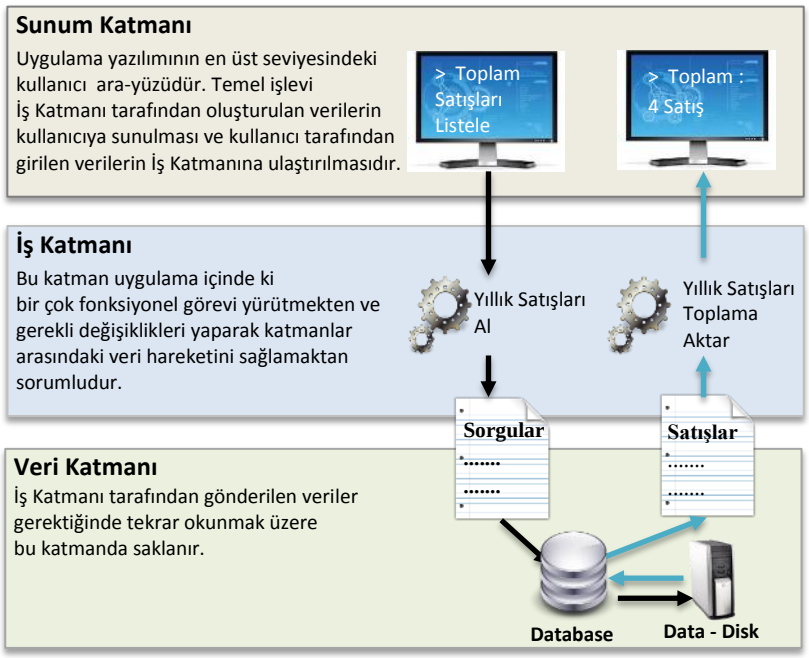
2.1 3-Katmanlı Mimari

3-Katmanlı mimari, uygulama yazılımının katmanlarını birbirinden ayıran mimari bir paterndir. Bu patern *sunum katmanını*, *iş katmanını* ve *veri katmanını* birbirinden ayırır. İş Yazılımları için sıklıkla kullanılan bu patern daha güvenli, esnek ve sürdürülebilir bir yazılım modeli sağlar [18, 19]. 3-Katmanlı yazılım modeline ait ana bileşenler aşağıda tanımlanmış ve Şekil 1'de gösterilmiştir.

- Veri Katmanı: İş katmanı tarafından gönderilen verilerin kaydedilmesini ve gerektiğinde kaydedilmiş verilerin tekrar okunmasını sağlar.

- İş Katmanı: verinin değiştirilmesi ve diğer katmanlar arasında (Veri Katmanı ve Sunum Katmanı) taşınmasından ve genel olarak fonksiyonel işlevlerin yerine getirilmesinden sorumlu katmandır.

- Sunum Katmanı: Verinin kullanıcıya sunulması ve kullanıcı tarafından girilen verilerin İş Katmanına ulaştırılması için bir ara yüz sağlar.



Şekil 1. Üç-Katmanlı Mimari.

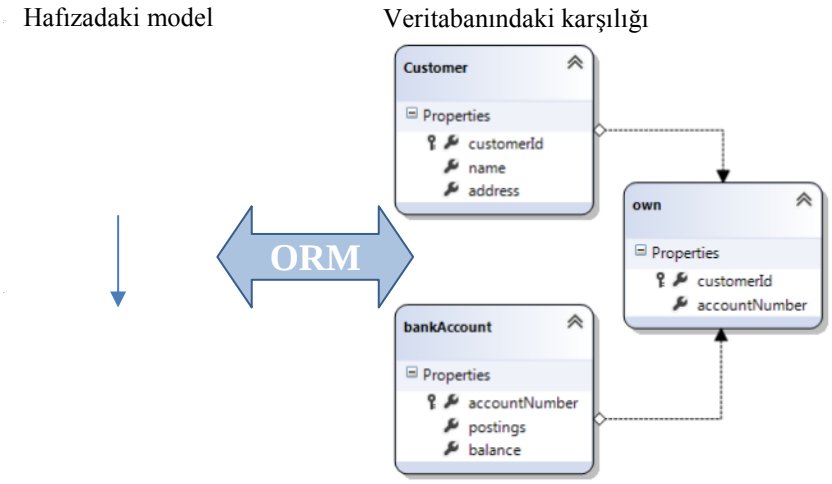
2.2 Nesne İlişkisel Eşleme

ORM (Object Relational Mapping) olarakta bilinen Nesne-İlişkisel Eşleme, Nesne-Tabanlı Analiz/Dizayn ve Nesne-Tabanlı Programlamada sıklıkla kullanılan ve uygulama yazılımına ait miras ile devralma yeteneğine sahip veri nesnelerinin

(sınıfların), veritabanlarındaki iki boyutlu tablolara aktarılmasını sağlayan bir tekniktir [20]. Bu teknik, veritabanı ulaşım protokollerinden ve kurulacak olan veritanı yapısından bağımsız olarak verinin kaydedilmesine ve gerektiğinde tekrar okunmasına olanak sağlar. Nesne-İlişkisel Eşlemenin temel amacı; Nesne-Tabanlı programlama tekniği ile oluşturulan ve miras ile devralma yeteneğine sahip nesne yapısının veri tablolarına en doğru şekilde aktarılmasıdır.

Nesne-İlişkisel yapılar ve bileşenlerden, Java geliştirme ortamı için *The Grails Framework* [21] ve *Hibernate* [22], ASP.NET geliştirme ortamı için *NHibernate* [23] ve Android geliştirme ortamı için *ActiveAndroid ORM* [24] yazılımcılar arasında popüler olanlardan bazılarıdır.

Şekil 2 birbiri ile ilişkili iki iş nesnesinin (*Customer* ve *BankAccount*) Nesne-İlişkisel Eşleme metodu kullanılarak veri tabanına nasıl kaydedildiğini göstermektedir. Bu metod veritabanı erişim protokollerinden bağımsız bir şekilde, kullanıcıdan herhangi bir girdi almadan otomatik olarak gerekli olan veri tablosu yapısını ve ilişkisel anahtarları oluşturacak ve ardından veriyi kaydedecektir..



Şekil 2. İki iş nesnesi için örnek bir Nesne-İlişkisel Model.

3 Eşleme ve Ölçüm Aracı

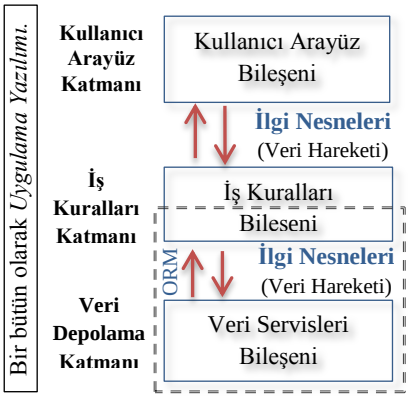
3.1 Eşleme

Şekil 3 ve Şekil 4'te görüldüğü gibi 3-Katmanlı iş yazılımlarının katmanları ile COSMIC tarafından tanımlanan iş uygulama yazılımlarına ait katmanlar arasında büyük bir benzerlik bulunmaktadır. Her iki yapıda da veri *ilgi nesneleri* olarak taşınmaktadır. COSMIC konseptinde verinin saklandığı kısım olarak değerlendirilen *Veri Depolama Katmanı* 3-Katmanlı iş uygulamasında *Veri Katmanına* denk gelmektedir. Benzer şekilde iş uygulamasında bulunan *İş Katmanı* ve *Sunum Katmanı* COSMIC konseptinde *İş Kuralları Katmanı* ve *Kullanıcı Arayüz Katmanına* kaşılık gelmektedir. Her iki yapıdaki iş katmanları genel olarak

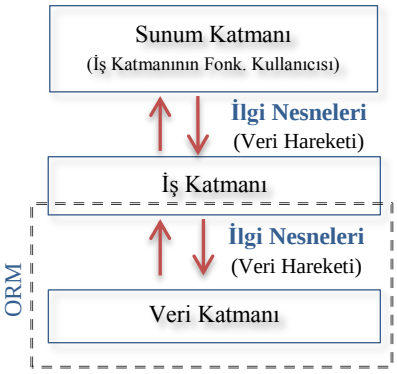
uygulamanın fonksiyonel kabiliyetlerini barındıran katmanlardır. Kullanıcı arayüz katmanı ve sunum katmanı ise verilerin kullanıcıya sunulmasını ve kullanıcı tarafından girilen verilerin iş katmanlarına aktarılmasını sağlamaktadır. COCMIC konseptine ait tanımların, iş uygulama yazılımlarında nelere denk geldikleri aşağıda detaylı bir şekilde listelenmiştir.

Uygulama Katmanı: COSMIC konseptinde uygulama katmanı; *Kullanıcı Arayüz katmanı, İş Kuralları Katmanı* ve *Veri depolama Katmanını* içeren bir katman olarak tanımlanmaktadır [6]. Çalışmamızdaki eşlemede 3-Katmanlı uygulama yazılımının temel parçaları olan *Sunum, İş* ve *Veri* Katmanları, COSMIC konseptindeki bu katmanlara denk gelmektedir. Genel olarak ifade etmek gerekirse COSMIC'te tanımlanan *Uygulama Katmanı* 3-Katmanlı mimaride İş Uygulamasını temsil etmektedir.

Uygulama Sınırları: COSMIC Genel Yazılım Modelinde uygulama sınırları; uygulamayı çevreleyen konseptsel bir sınır olarak tanımlanmaktadır [6]. 3-Katmanlı yazılım mimarisine göre ise sunum katmanı, veri işleme kabiliyetine sahip olmadığı ve sadece verinin/olayların taşınması ve sunulmasını sağlayan bir katman olması sebebi ile "ince" bir katman olarak tanımlanmaktadır. Bu yüzden COSMIC konseptindeki uygulama sınırları, 3-Katmanlı iş uygulamalarında arayüz kontrollerini ve davranışlarını (Grafik Kullanıcı Arayüzü, Web Servis Arayüzü vb.) taşıyan sunum katmanına denk gelmektedir.



Şekil 3. COSMIC konseptinde uygulama yazılımına ait üç katman [6].



Şekil 4. Nesne-İlişkisel Eşleme metodu kullanan 3-Katmanlı İş Uygulama yazılımına ait katmanlar [6].

İşlevsel kullanıcılar: Bir yazılım parçasının İşlevsel Kullanıcı Gereksinimleri verisinin göndereni yada alıcısı olan kullanıcı (tipi) [6].

İlgi nesnesi: İşlevsel Kullanıcı Gereksinimleri bakış açısından belirlenen herhangi bir 'şey'. Fiziksel bir şey olabileceği gibi, işlevsel kullanıcı dünyasında yazılımın veriyi işlemesi ve/veya saklaması gereken kavramsal bir nesne veya bir kavramsal nesnenin parçası olabilir [6].

Veri Grubu: Bir veri grubu her bir veri özniteliğinin aynı ilgi nesnesinin tamamlayıcı özelliklerini tanımladığı farklı, boş olmayan, sıralı olmayan ve tekrarlamayan bir veri öznitelikleri setidir [6].

İşlevsel Süreç (İS): Bir işlevsel süreç benzersiz, kohesif ve bağımsız olarak çalıştırılabilen bir set veri hareketini içeren bir set İşlevsel Kullanıcı Gereksiniminin temel bileşenidir. Yazılım parçasının işlevsel olay belirlediği bilgisini veren işlevsel kullanıcılardan gelen veri hareketi (bir Giriş) ile tetiklenir. İşlevsel olaya karşılık olarak gereken işlemler yapıldığında tamamlanır [6].

Tetikleyici Olay (TO): Bir yazılım parçasının bir işlevsel kullanıcısının bir veya birden fazla işlevsel süreç başlatmasına ('tetikleme') yol açan bir olay (gerçekleşen bir şey) olarak tanımlanır. Bir set İşlevsel Kullanıcı Gereksiniminde, işlevsel kullanıcının bir işlevsel süreç tetiklemesini sağlayan her bir olay:

- Adı geçen İKG seti için alt parçalara bölünemez ve
- Ya olmuş veya olmamıştır [6].

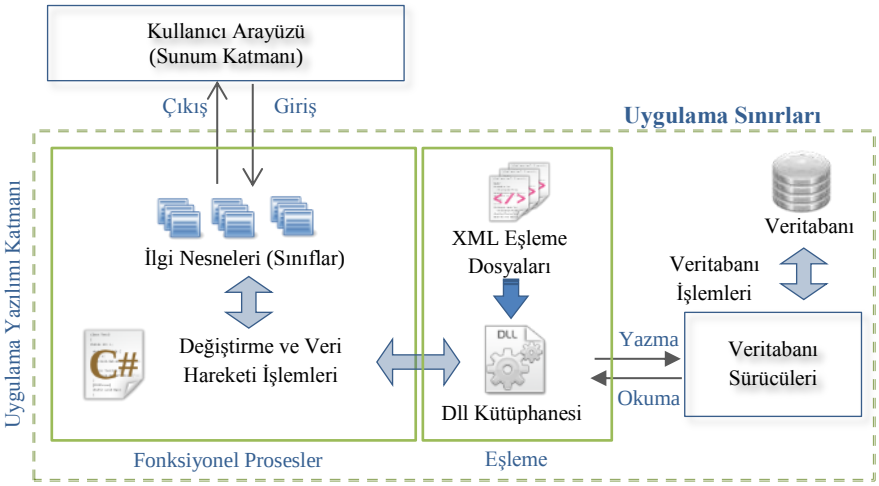
Veri Hareketleri:

- Giriş (G): Bir veri grubunu bir işlevsel kullanıcıdan uygulama sınırından içeriye ihtiyaç duyulan bir işlevsel sürece doğru hareket ettiren veri hareketidir [6].
- Çıkış (Ç): Bir veri grubunu bir işlevsel süreçten uygulama sınırından dışarıya ihtiyaç duyan bir işlevsel kullanıcıya doğru hareket ettiren veri hareketidir [6].
- Okuma (O): Bir veri grubunu kalıcı bellekten ihtiyaç duyulan işlevsel sürece doğru hareket ettiren bir veri hareketidir [6].
- Yazma (Y): Bir işlevsel süreçte yer alan veri grubunu kalıcı belleğe doğru hareket ettiren veri hareketidir [6].

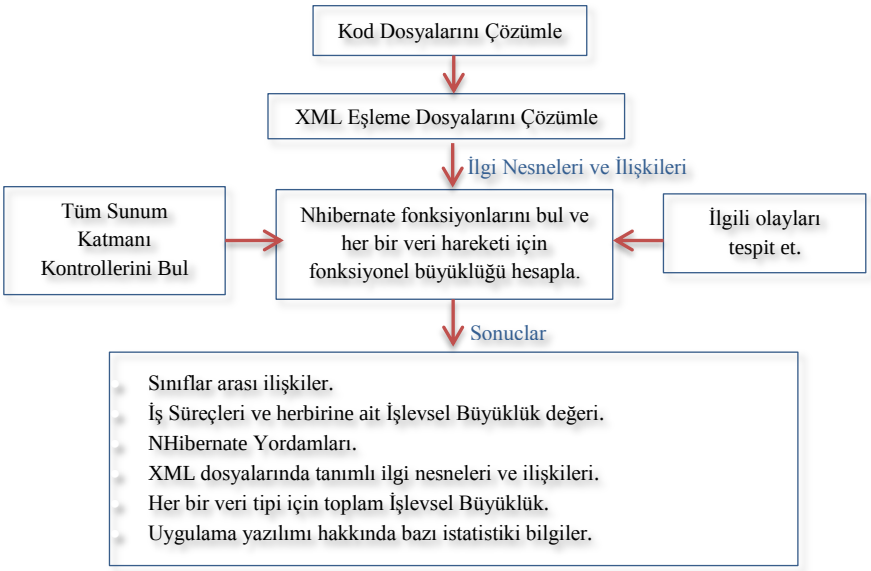
3.2 Ölçüm Aracı

Bu çalışmada sunulan otomatik ölçüm aracı; bölüm 3.1'de belirtilen eşleme yöntemine göre tasarlanmış, ASP.NET ortamında implement edilmiş ve *Nhibernate* Nesne-İlişkisel Eşleme kütüphanesini [23] kullanan iş uygulamalarını ölçmek üzere geliştirilmiştir. Ölçüm aracı *Visual Studio 2012 Express Edition* geliştirme ortamında c# proglama dili kullanılarak yazılmıştır.

Şekil 5 ölçüm aracı perspektifinden ölçülecek olan yazılıma ait; ilgi nesneleri seviyesinde genel veri akış sürecini, uygulamanın temel yazılım parçacıklarını, konfigürasyon/eşleme dosyalarını ve veri tabanı ile ilişkisini göstermektedir.



Şekil 5. Ölçülecek olan yuygulama yazılımına ait genel bir veri akış süreci.



Şekil 6. Ölçüm aracına ait ölçüm prosesi.

Ölçülecek olan yazılıma ait genel tanımları yaptıktan sonra artık ölçüm aracının ölçüm sürecini nasıl işlettiğini açıklayabiliriz. Şekil 6'da gösterildiği gibi ölçüm süreci temel olarak 5 bölümden oluşmaktadır. Ölçüm, iş uygulama yazılımına ait dosyaların ve *XML Eşleme* (bkz. Şekil 5) dosyalarının çözümlenmesi ile başlamaktadır. Ardından *XML Eşleme* dosyaları yardımı ile *iş nesneleri* tespit edilmekte, sonrasında ise kod dosyaları kullanılarak işlevsel süreçlerin

gerçekleştirilmesini sağlayan yordamlar ve birbirleri ile ilişkileri tespit edilmektedir. Gerekli bileşenlerin tespitinin tamamlanmasının ardından çalışan ana fonksiyon; ilgi nesneleri, yordamlar ve hem iş nesnelerinin birbirleri ile olan miras bağlarını hemde yordamların birbirleri ile olan çağırma ilişkilerini takip ederek her bir işlevsel süreç için toplam veri hareketi miktarını COSMIC kurallarına göre hesaplamaktadır. Sonuç olarak ise Şekil 6'da *sonuç* bölümünde belirtilen veriler kullanıcıya sunulmaktadır.

Ölçüm aracının implementasyonunda; ölçülecek olan yazılımdaki sınıf ve yordam gibi tanımlamaların bulunması ve aralarındaki ilişkilerin tespit edilmesi amacı ile *Nova.Codedom* [25] olarak adlandırılan bir kütüphane kullanılmıştır. Ölçüm ile ilgili olarak veri analizine başlamadan önce deneysel çalışmamızda, ölçmek amacı ile kullandığımız MVC (Model View Control) uygulamasına ait Şekil 7'de gösterilen veri akış şemasının ve COSMIC konseptindeki karşılıklarının belirlenmesi gerekmektedir. Şekil 7'de 1, 8, 10-5 ve 6 hareket numarası ile tanımlanan veri harketleri sırası ile Bölüm 3.1'de de tanımlanan ***G**iriş*, ***Ç**ıkış*, ***Y**azma* ve ***O**kuma* veri hareketlerine denk gelmektedir. Diğer hareketler (2, 3, 4, 7 ve 9) ise verinin işlenmesi ile ilgili süreçler olarak değerlendirildiğinden COSMIC Manueline [6] göre veri hareketi olarak değerlendirilmemektedir.

4 Deneysel Çalışma

Ölçüm ile ilgili olarak sunulan yaklaşımın ve ölçüm aracının teset edilmesi amacı ile deneysel bir çalışma yapılmış ve elde edilen otomatik ölçüm sonuçları, manuel yordamlarla elde edilen sonuçlar ile karşılaştırılmıştır.

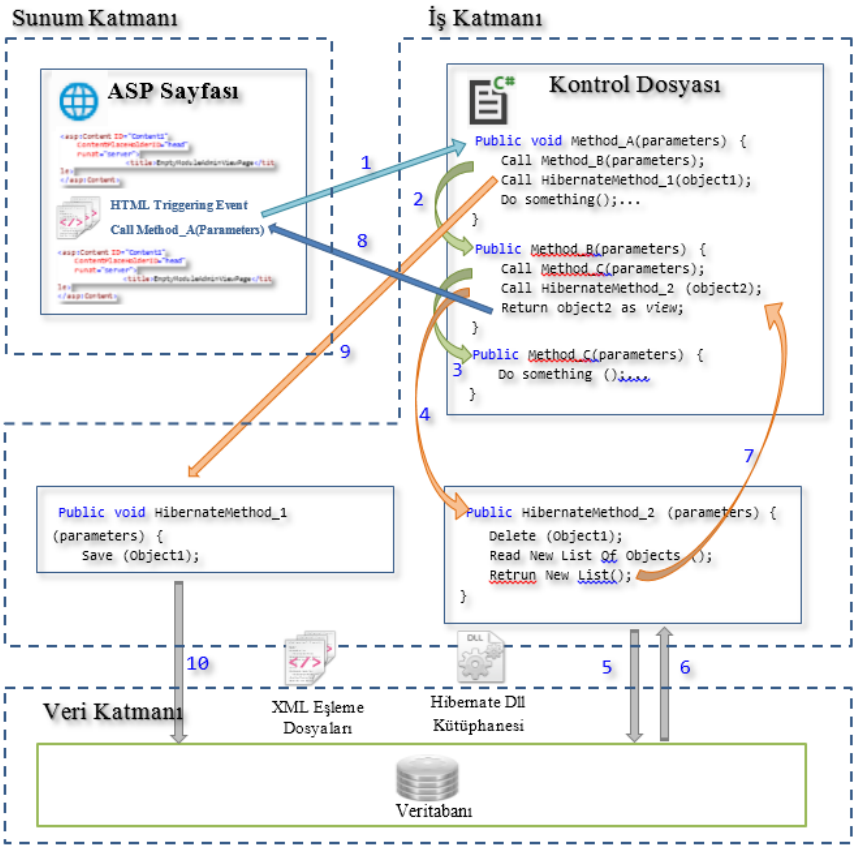
4.1 Seçilen İş Uygulaması

Ölçülecek yazılım olarak; *Cuya-hoga* adında açık kaynaklı ve 3-Katmanlı Nesne İilişkisel Eşleme metodu kullanan bir MVC web portalı yönetim yazılımı seçilmiştir [26]. Bu iş uygulaması Visual Studio geliştirme ortamında c# programlama dili kullanılarak geliştirilmiş olup, 9 Visual Studio Projesinden[1] ve 16744 Kaynak Kod Satırından (SLOC: Source Lines of Code) oluşmaktadır.

4.2 Veri Toplama

Cuya-Hoga uygulamasına ait bir İşlevsel Kullanıcı Gereksinim dokümanı olmadığı için bu gereksinimler uygulama yazılımının kaynak kodundan ve ürün tanıtımı için kullanılan web sayfasından çıkarılmıştır [26]. Manuel yordam ile yapılan ölçümler COSMIC ölçüm sertifikasına sahip bir uzman ölçümcü ve bu makalenin yazarı tarafından gerçekleştirilmiştir. Manuel ölçüm için toplamda 29 saat harcanmıştır.

[1] Visual Studio projesi iş uygulama yazılımının modülleri olarak düşünülebilir. Bu modüller uygulamanın çalışabilmesi için gereken tüm dosyaların her projede ayrı ayrı gruplanabilmesine imkan sağlar.

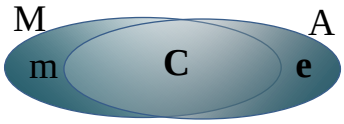


Şekil 7. MVC uygulama yazılımına ait detaylı veri akış şeması.

4.3 Veri Analizi ve Sonuçlar

Bu çalışmada, birbiri ile karşılaştırılan üç tip veri grubu oluşturulmuştur. Bu veri tiplerinden birincisi Manuel ölçüm (M) sonuçları, ikincisi ölçüm aracı tarafından bulunan Otomatik (A) ölçüm sonuçları, sonuncusu ise bu iki sonuç kümesinin kesişimi olan Ortak Sonuç (C) kümesidir. Çalışmada ki analiz ve değerlendirmeler yapılırken manuel ölçüm sonuçları (M) tam doğru olarak kabul edilecek ve hesaplamalarda %100 olarak değerlendirilecektir. Tanımlanan bu üç temel veri tipine ek olarak; ölçüm aracı tarafından tespit edilemeyen ancak ölçümcüler tarafından bulunan sonuçlar (m) ile araç tarafından veri hareketi olarak değerlendirilen ancak ölçümcü tarafından tepit edilemeyen yada veri hareketi olarak değerlendirilmeyen (e) sonuçlarda hesaplamalarda kullanılmıştır. Ayrıca her bir fonksiyonel proses için manuel ve otomatik ölçümler karşılaştırılmış ve yanlış

bulunan veri hareketleri[2] çıkarılarak sonuçlar doğrulandıktan sonra hesaplamalara dahil edilmiştir. Şekil 8 yukarıda bahsedilen veri tipleri ve aralarındaki ilişkiyi göstermektedir.



Şekil 8. Bu deneysel çalışmada kullanılan veri tiplerinin sınıflandırılması.

Manuel yordamlar (M) ve ölçüm aracı kullanılarak elde edilen sonuçlar (A) ile bu iki küme kullanılarak hesaplanan veriler (C, m ve e) Table 1'de verilmiştir. Toplamda, manuel yöntemlerle 410 veri hareketi ve 86 işlevsel süreç bulunmuş olup, bu sonuçlara karşılık olarak ölçüm aracı tarafından gerçekletirilen ölçüm işlemi ile 385 veri hareketi ve 93 işlevsel süreç tespit edilmiştir. Yapılan değerlendirmelerde kullanılan yakınlık yüzdeleri *C*100/M* denklemi ile hesaplanmıştır. Ortak Sonuç Kümesi olan C değeri otomatik (A) yada manuel (M) sonuçlar kullanılarak *C=A-e* yada *C=M-m* eşitlikleri ile hesaplanmaktadır. Bölüm 4.4'te değerlendirilen manuel ve otomatik ölçüm sonuçları; her bir veri tipi için ayrı ayrı, toplam veri hareketleri ve işlevsel süreç sayıları olarak Table 2 verilmiş ve Fig. 9 ve Fig. 10'de gösterilmiştir.

Table 1. Manuel ve Otomatik ölçüm sonuçları.

	Toplam G	Toplam Y	Toplam O	Toplam Ç	Toplam Veri Hareketi	Toplam İS
M:	71	111	140	88	410	86
A:	39	116	167	63	385	93
m:	33	5	5	25	68	11
e:	5	14	28	4	51	18
C:	34	102	139	59	334	75
Yakınlık:	**47.89**	**91.89**	**99.29**	**67.05**	**81.46**	**87.21**

[2] Otomatik ölçüm sonucunda bulunan bir veri hareketinin manuel ölçümde de olduğunun teyidi yapılmış ve fazla veya eksik çıkan otomatik ölçüm sonuçları hesaplamalara negatif olarak dahil edilmiştir.

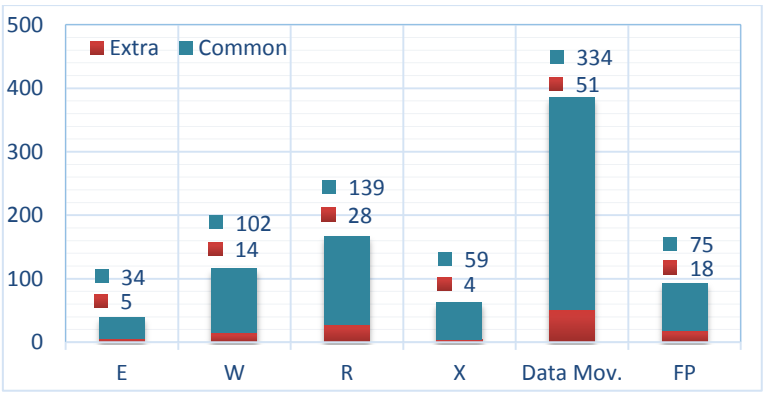


Fig. 9. Her bir veri hareketi için toplam otomatik sonuçlar ile C ve e değerleri.

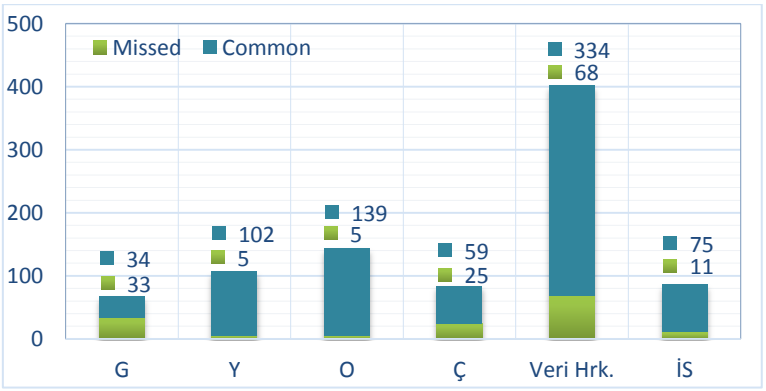


Fig. 10. Her bir veri hareketi için toplam manuel sonuçlar ile C ve m değerleri.

4.4 Sonuçların Değerlendirilmesi

G, Y, O ve Ç veri hareketleri için manuel ve otomatik ölçüm sonuçları sırası ile 71, 111, 140, 88 ve 39, 116, 167, 63 olarak tespit edilmiş ve toplam veri hareketi yakınsama oranı bölüm 4.3'te belirtilen yöntemle %81.46 olarak hesaplanmıştır. Elde edilen otomatik sonuçlar ortalama bir masa üstü bilgisayar ile bir dakikadan daha az bir zamanda bulunmuştur. Table 1, Fig. 9 and Fig. 10'da verilen sonuçlardan da anlaşılacağı gibi Y ve O veri hareketlerinde elde edilen yakınsama oranları, G ve Ç veri hareketlerinde elde edilen yakınsama oranlarından daha yüksektir. Bunu sebebi; ölçülen yazılımda Y ve O veri hareketleri ile ilgili işlemlerin *Nhibernate* Nesne İlişkisel Eşleme kütüphanesi tarafından kullanılan standart metodlarla sağlanmasına karşın, G ve Ç veri hareketlerinin programlayıcının kullandığı stillere bağlı olarak değişiklik gösteren farklı grafik arayüz implementasyonlarına dayanması gösterilebilir. Standart olarak kullanılan yöntemlerin tespit ve takip edilmesi daha kolay olduğu için Y ve O veri hareketlerinin ölçüm aracı tarafından bulunma olasılıkları daha yüksektir.

4.5 Doğrulama Tehditleri

Bu çalışmada sunulan yaklaşım; Nesne İlişkisel Eşleme metodu kullanan ve *ilgi nesneleri* ilişkilerini XML eşleme dosyaları vasıtasıyla tanımlayan neredeyse tüm iş yazılımlarına uyarlanabilir. Fakat halihazırdaki ölçüm aracına sadece c# programlama dili implement edilmiştir. Bu yüzden ölçüm yapılan uygulamalarda kısmen de olsa diğer programlama dillerinin kullanılması (java vb.) ölçüm aracının veri hareketlerini tespit kabiliyetini kısıtlamaktadır. Buna ek olarak; ölçüm aracı geliştirilme aşamasında bazı küçük uygulamalar üzerinde test edilmiş olsada, sadece bir büyük proje üzerinde denenmiştir.

Cuya-hoga uygulamasının manuel büyüklük ölçümü, İşlevsel Kullanıcı Gereksinimleri için kullanılabilecek bir doküman olmadığından kod ve arayüzden yapılmıştır. Bu yüzden ölçüm için harcanan süre 29 saat olarak ölçülmüş ancak bu sürenin normalden fazla olduğu değerlendirilmiştir. İşlevsel Kullanıcı Gereksinim dokümanlarının olması durumunda bu sürenin biraz daha azalabileceği değerlendirilmiştir.

Yakınsama yüzdelerinin hesaplanmasında otomatik (A), manuel (M) ve ortak veri seti (C) kullanılmıştır. Ancak normal koşullarda ölçüm aracını kullanacak olan ölçümcü sadece otomatik olarak üretilen sonuçları görebilecektir. Buna göre hesaplanacak bir yakınsama, *A*100/M* formülü ile hesaplanacağından bu çalışma için alınacak yakınsama sonucu Table 2'de de gösterildiği gibi %93.90 olacaktır. Her nekadar bu çalışma için daha yakınsak bir sonuç elde edilmiş olsada bunun her zaman doğru almayacağı açıktır.

Table 2. Otomatik ve Manuel ölçümlerin karşılaştırılması.

	Toplam E	**Toplam W**	**Toplam R**	**Toplam X**	**Toplam Hareket**
M:	71	111	140	88	410
A:	39	116	167	63	385
C:	34	102	139	59	334
A için Yakınsama :	**54.92**	**95.49**	**80.71**	**71.59**	**93.90**
C için Yakınsama :	**47.89**	**91.89**	**99.29**	**67.05**	**81.46**

5 Sonuç

Bu çalışma ile 3-Katmalı Nesne İlişkisel Eşleme metodunu kullanan iş uygulamalarının ölçümü için bir yaklaşım sunulmuş, bu yaklaşımı uygulayabilecek prototip bir ölçüm aracı geliştirilmiş ve sunulan yaklaşım deneysel bir çalışma ile doğrulanmıştır. Table 2, Fig. 9 ve Fig. 10'de sunulan sonuçlardan da görülebileceği gibi implement edilen ölçüm aracı, proje yöneticisine yazılım büyüklük ölçümü için güvenilir kabul edilebilecek ölçüm sonuçlarını kısa bir zaman içerisinde uzman bir ölçümcüye ihtiyaç duymadan sunmaktadır. Ayrıca araç tarafından otomatik ölçüm sonuçlarına ek olarak dosya sayıları ve tipleri, ilgi nesneleri ve aralarındaki ilişkiler, tetikleyici olaylar ve bağlı oldukları arayüz kontrolleri, yordamlar ve her bir İS için gömülü alt yordamlar ve bu yordamlarla kullanılan parametreler gibi daha bir çok bilgi ölçümcüye sunulmaktadır. Bu yetenek ölçümcünün, ölçüm aracı tarafından yanlış değerlendirilmiş veri hareketlerini düzeltmesi ve ölçüm sonucunu istediği gibi

geliştirebilmesi için imkan sunmaktadır. Ayrıca otomatik ölçüm kod üzerinden yapıldığı için eksik yada güncel olmayan İKG dokümanları nedeniyle ölçülemeyen uygulamaların ölçülebilmesinede imkan sağlamaktadır.

Ayrıca aynı mimariye sahip yazılım tipleri bu ölçüm aracı tarafından ek bir çalışma ihtiyacı olmadan tekrarlı olarak ölçülebilirler. Buna ek olarak; bu ölçümler çok kısa bir zaman içerisinde çok az bir eforla yapılabilirler. Ayrıca ölçüm aracı tarafından yapılan ölçümler -en azından işlevsel büyüklük hakkında ön bir bilgi elde etmek amacı ile- uzman bir ölçümcüye ihtiyaç duyulmadan gerçekleştirilebilirler.

Yukarıda bahsedilen ajantajlar yanında ölçüm aracının; tasarımında yanlış implement edilmiş tespit algoritmalarının bulunma olasılığı, bazı implementasyonlarda veri hareketlerini yanlış değerlendirmesi yada tespit edememesi, tüm İS'leri bulamaması veya gömülü İS'leri takip edememesi gibi bazı dezavantajları ve eksiklikleride bulunmaktadır. Bu deneysel çalışmada ölçüm aracı sadece bir adet orta büyüklükte iş uygulama yazılımının ölçümünden elde edilen sonuçların karşılaştırılması ile değerlendirilmiştir. Buda, ölçüm performansı konusunda ki değerlendirmenin doğruluğunun ileride yapılacak başka çalışmalar yardımı ile desteklenmesini gerektirmektedir.

Bölüm 3.1 ve 3.2'de belirtildiği üzere bu ölçüm aracı sadece standart yordamlar kullanılarak yapılan implementasyonlarda başarı sağlayabilmektedir. Programcının standandart yollardan ayrılması ölçüm aracının tespit kabiliyetini azaltacak ve sonuçların yakınsama oranlarını düşürecektir.

Bu araç, c# programlama dili kullanılarak implement edilmiş olan, XML dosyaları vasıtası ile ilgi nesnelerini tanımlayan ve Nesne-İlişkisel Eşleme metodu kullanan 3-Katmanlı iş uygulama yazılımları için geliştirilmiş olup, sonraki çalışmalarda ilgi nesnelerinin farklı şekillerde tanımlandığı (örn.: Fluent Mapping[3]) ve diğer programlama dilleri (örn.:Java) kullanılarak implement edilmiş daha büyük ölçekli endüstriyel uygulamaların ölçümü için iyileştirmeler planlanmaktadır.

[3] Fluent mapping; ilgi nesnelerinin XML dosyaları yerine kod içerisinde kullanılan terimler yardımı ile yapıldığı bir Eşleme metodudur.

Kaynaklar

[1] G. Yılmaz, S. Tunalılar and O. Demirörs, "R-COVER: Yazılım Büyüklük Ölçümü Hata Tespit Aracı," UYMS-2013, Ankara, 2013.

[2] R. Meli, A. Abran, T. Ho Winh and S. Oligny, "On The Applicability of COSMIC-FFP For Measuring Software Throughout Its Life Cycle," in *11th European Software Control and Metrics Conference*, Munich, Germany, 2000.

[3] O. Turetken, O. O. Top, B. Ozkan and O. Demirors, "The Impact of Individual Assumptions on Functional Size Measurement," in *International Conferences IWSM 2008* , Munich, Germany, 2008.

[4] A. Stambollian and A. Abran, "Survey of Automation Tools Supporting COSMIC-FFP – ISO 19761," École de Technologie Supérieure-ÉTS, Montréal (Québec) Canada.

[5] S. Trudel, "Using The Cosmic Functional Size Measurement Method (ISO 19761) As A Software Requirements Improvement Mechanism," École De Technologie Supérieure, Montreal, 2012.

[6] A. Abran, J.-M. Desharnais, S. Oligny, D. St-Pierre and C. Symons, The COSMIC Functional Size Measurement Method (Version 4.0.1), COSMIC, 2015.

[7] P. Fagg, A. Lesterhuis, L. Santillo and F. Vogelezang, Guideline for Sizing Service-Oriented Architecture Software, The Common Software Measurement International Consortium (COSMIC), 2010.

[8] B. Marin, N. Condori-Fernandez, O. Pastor and A. Abran, "Measuring the Functional Size of Conceptual Models in an MDA Environment," in *CAiSE'08 Forum*, Montpellier, France, 2008.

[9] A. A. Akca and A. Tarhan, "Run-time measurement of COSMIC functional size for Java business applications: Is it worth the cost.," Ankara/Turkey, 2013.

[10] A. J. Albrecht, Measuring Application Development Productivity, IBM Corporation, 1979.

[11] K. v. d. Berg, T. Dekkers and R. Oudshoorn, "Functional size measurement applied to UML-based user requirements," in *2nd Software Measurement European Forum, SMEF 2005*, Rome, Italy , 2005.

[12] "Estimating Software Size with UML Models," in *C3S2E 2008*, New York, 2008.

[13] V. T. Ho and A. Abran, "A Framework For Automatic Function Point Counting From Source Code," in *International Workshop on Software Measurement (IWSM'99)*, Lac Superieur-Canada, 1999.

[14] B. Marin, O. Pastor and G. Giachetti, "Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment," Department of Information Systems and Computation Technical Univercity of Valrncia, Valencia Spain, 2011.

[15] K. Paton, "Automatic Function Point Counting Using Static And Dynamic Code Analysis," in *International Workshop on Software Measurement (IWSM'99)*, Canada, 1999.

[16] H. Soubra, A. Abran, S. Stern and A. R. Cherif, "Design of a Functional Size

Measurement Procedure for Real-Time Embeded Software Requirements Expressed using the Simulink Model," in *International Workshop on Software Measurement* , Nara, Japan, 2011.

[17] D. Hassan, M. Frappier and R. St-Denis, "A Formal Definition of COSMIC-FFP for Automated Measurement of ROOM Specifications," in *Proc. 4th Eur. Conf. Software Measurement and ICT Control*, Heidelberg, 2001.

[18] R. N. Taylor, N. Medvidovic and E. M. Dashofy, Software Architecture Foundations, Theory, And Practice, USA: John Wiley & Sons Inc., 2010.

[19] A. Aarsten, D. Brugali and G. Menga, "Patterns for Three-Tier Client/Server Applications," Torino-Italy, 1996.

[20] M. Keith and M. Schnicariol, "Object-Relational Mapping," Apress, 2010.

[21] Grails, "A powerful Groovy-based web application framework for the JVM," The Grails, 2015. [Online]. Available: https://grails.org/index.html. [Accessed 06 05 2015].

[22] G. Rocher, P. Ledbrook, M. Palmer, J. Brown, L. Daley and B. Beckwith, "GORM," Spring Source, [Online]. Available: http://grails.org/doc/2.3.x/guide/GORM.html. [Accessed 24 03 2014].

[23] J. Dentler, NHibernate 3.0 Cookbook, Birmingham, UK: Packt Publishing Ltd., 2010.

[24] N. Esquenazi, "ActiveAndroid Guide," 09 03 2015. [Online]. Available: https://github.com/codepath/android_guides/wiki/ActiveAndroid-Guide. [Accessed 04 05 2015].

[25] Inevitable-Software, "C# Parser and CodeDOM," Inevitable Software Inc., 2013. [Online]. Available: http://www.inevitablesoftware.com/. [Accessed 06 05 2015].

[26] C. Hoga, "Cuya Hoga," Cuya Hoga Project Team, [Online]. Available: http://cuyahoga-project.org/. [Accessed 04 08 2014].