Journal of the Faculty of Engineering and Architecture of Gazi University 32:3 (2017) 887-900

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Açık kaynak kodlu yazılımların problem ve çözüm alanı ölçüleri arasındaki ilinti

Tülin Erçelebi Ayyıldız[1*], Altan Koçyiğit[2]
[1]Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü, 06790, Ankara, Türkiye
[2]Orta Doğu Teknik Üniversitesi, Bilişim Sistemleri Bölümü, 06810, Ankara, Türkiye

ÖNEÇIKANLAR

- Açık kaynak kodlu yazılımların problem ve çözüm alanı ölçüleri arasındaki ilinti
- Açık kaynak kodlu yazılım projelerinde aykırı değer analizi
- Doğrusal bağlanım tekniği ile yazılım büyüklük ölçümü

Makale Bilgileri
Geliş: 15.12.2015
Kabul: 23.06.2017

DOI:
10.17341/gazimmfd.337641

Anahtar Kelimeler:
Yazılım büyüklük ölçümü, problem alanı, çözüm alanı, doğrusal bağlanım, açık kaynak kodlu yazılım projeleri

ÖZET

Halen kullanılmakta olan yazılım büyüklüğü ölçümü ve efor kestirimleri genellikle geliştirilecek olan yazılımın detaylı gereksinimlerini temel girdi olarak kullanırlar ve büyüklük ölçümü için belirli bir zamana ve uzmanlığa ihtiyaç duyarlar. Bu çalışma, açık kaynak kodlu yazılımların problem alanı ölçüleri (isim ve fiil sayıları) ile çözüm alanı ölçüleri (yazılım sınıf ve yordam sayıları) arasındaki ilintiyi incelemektedir. Makalede 27 açık kaynak kodlu yazılım projesi değerlendirilmiştir. Problem alanının (kavramsal) büyüklüğü ile çözüm alanının (tasarım) büyüklüğü arasındaki ilintiyi incelemek için doğrusal bağlanım tekniği uygulanmıştır. Sonuçlar problem alanı ölçüleri ile ilgili yazılım bileşenlerini oluşturan çözüm alanı ölçüleri arasında güçlü ilintiyi göstermektedir. Sonuçlar, yazılım geliştirme projelerinin erken safhalarındaki problem alanı tanımlamalarını kullanarak makul büyüklük ve efor kestirimleri yapılabilmesinin mümkün olduğuna işaret etmektedir.

Correlations between problem and solution domain measures of open source software

HIGHLIGHTS

- Correlations between problem and solution domain measures for open source software
- Outlier analysis for open source software
- Software size measurement with using linear regression technique

Article Info
Received: 15.12.2015
Accepted: 23.06.2017

DOI:
10.17341/gazimmfd.337641

Keywords:
Software size measurement, problem domain, solution domain, linear regression, open source software projects

ABSTRACT

Software size measurement and effort estimation methodologies in use today usually take the detailed requirements of software to be developed as the primary input and a certain amount of time and expertise is needed for size measurement. This paper analyzes the open source projects' correlations between the problem domain measures (the number of nouns and verbs) and solution domain measures (the number of software classes and methods). In this paper, 27 open source software projects are analyzed. Linear regression and cross validation techniques are applied to investigate the relation between the sizes of problem domain (i.e., conceptual) and solution domain (i.e., design) measures. The results reveal a strong correlation between the problem domain measures and the solution domain measures constituting the corresponding software. The results suggest that it is possible to use problem domain descriptions in the early stages of software development projects to make plausible predictions for the size and effort of the software.

*Sorumlu Yazar/Corresponding Author: ercelebi@baskent.edu.tr / Tel: +90 312 246 66 61

1. GİRİŞ (INTRODUCTION)

Yazılım büyüklüğü, yazılım geliştirme maliyeti ve süresi kestiriminde, başarım ve kalite ölçümünde, risk değerlendirmede, üretkenlik ölçümünde, sözleşme yönetimininde ve bunlar gibi birçok farklı amaçla kullanılmaktadır. Bu nedenle, yazılımın büyüklüğünü olabildiğince erken ve asgari efor harcayarak belirlemek oldukça önemlidir. Literatürde birçok büyüklük tahmin yöntemi önerilmektedir. Örneğin, Živković ve arkadaşları [1] UML modelleri kullanarak İşlev Puanı yöntemiyle otomatik yazılım büyüklüğü tahmin yöntemi önermişlerdir. Azzeh ve arkadaşları [2] bulanık (fuzzy) model ağacı kullanarak erken safhada yazılım büyüklüğü ölçüm yöntemi önermişlerdir. Bir başka çalışmada ise Nassif ve arkadaşları [3] yapay sinir ağları yöntemiyle yazılım büyüklük ölçümüne dayalı yeni bir efor kestirim yöntem öne sürmüşlerdir. Bu yöntemler genellikle detaylı gereksinim dokümanlarını girdi olarak almaktadır ve daha önemlisi bu yöntemleri uygulamak ciddi miktarda zaman, maliyet ve uzmanlık gerektirmektedir. Yazılım büyüklük ölçümünde kullanılan yöntemler, teknik büyüklük ölçüm yöntemleri ve işlevsel büyüklük ölçüm yöntemleri olmak üzere iki kategoride incelenmektedir. Teknik büyüklük ölçüm yönteminde en çok kullanılan yöntem Satır Sayısı (Line of Code-LOC) yöntemidir. Bu yöntemde yazılan kaynak kodun satır sayısına bakılarak yazılım büyüklüğü ile ilgili ölçüm yapmak mümkündür. Kullanımı çok kolay olmasına rağmen, kişilerin kodlama tekniklerinin farklı olması veya kullanılan programlama dilinin değişkenlik göstermesi bu yöntemin dezavantajlarındandır. Bu nedenle İşlevsel Büyüklük Ölçümü (Functional Size Measurement -FSM) yöntemleri yazılımın büyüklüğünü ölçebilmek için daha çok tercih edilen yöntemler olmuşlardır. 1979 yılında satır sayısına alternatif olarak İşlev Puanı (Function Points -FP) yöntemi Albrecht tarafından öne sürülmüştür [4]. Daha sonra farklı birçok işlevsel büyüklük ölçümü yöntemi öne sürülmüştür.

Bunlardan bazıları şunlardır:

- IFPUG İşlev Puanı Analizi (IFPUG Function Point Analysis –IFPUG FPA),
- MARK II İşlev Puanı (MARK II Function Points –MK II FP),
- NESMA İşlev Puanı (NESMA Function Points),
- Tam İşlev Puanı (Full Function Points -FFP),
- COSMIC Tam İşlev Puanı (COSMIC Full Function Points –COSMIC FFP),
- Nesne Puanı (Object Points),
- Nesne Yönelimli İşlev Puanı (Object Oriented Function Points – OOFP)

Bu yöntemlerden MARK II FP, IFPUG FPA ve COSMIC FFP yöntemleri ISO (Uluslararası Standartlar Kurumu) standardı olarak kabul görmüştür. IFPUG FPA yöntemi Uluslararası İşlev Puanı Kullanıcıları Grubu tarafından 1979 yılında Albrecht'in öne sürdüğü İşlev Puanı yöntemine açıklık getirmek amacıyla çeşitli örnekler hazırlanarak geliştirilmiştir [5]. MARK II İşlev Puanı yöntemi Symons tarafından 1988 yılında IFPUG FPA yönteminin eksikliklerini gidermek amacıyla önerilmiştir [6]. Bu iki yöntem daha çok bilişim sistemlerinin işlevsel büyüklüklerini ölçmek için kullanılırken, 1999 yılında COSMIC FFP yöntemi öne sürülerek gerçek zamanlı sistemler için de işlevsel büyüklüğün ölçülmesi sağlanmıştır [7]. Nesne Puanı yöntemi, İşlev Puanı yöntemiyle çok benzerdir, fakat işlevlerin yerine nesneler sayılmaktadır. Nesneler, 3. nesil programlama dilleri modüllerini, raporlarını ve ekranlarını içermektedir [8]. Nesne Tabanlı İşlev Puanı yöntemi ise nesne tabanlı yazılım geliştirme projelerinde kullanılmak üzere geliştirilmiştir [9].

1993 yılında Karner'ın [10] öne sürdüğü Kullanım Durumu Puanı (Use Case Point) yöntemi ile de yazılım büyüklüğü sistemdeki aktör ve kullanım durumlarının sınıflandırılmasıyla elde edilmiştir. Yöntem ayrıca yazılım büyüklüğünü hesaplayabilmek için teknik karmaşıklık faktörü ve çevresel karmaşıklık faktörlerini de girdi olarak almaktadır. Sınıf Puanı (Class Point) yöntemi, tasarım dökümanını temel alarak nesne tabanlı yazılımların büyüklüklerini tahmin etmek için kullanılmaktadır. Yöntemde yazılım içerisindeki sınıflar belirlenerek sınıflandırılmaktadır. Belirlenen her bir sınıf için karmaşıklık düzeyi belirlenmektedir. [11]. Öte yandan, dilbilim yöntemlerinin nesne yönelimli yazılım geliştirme uygulamaları için kullanılması ilk defa 1983 yılında Abbott [12] tarafından önerilmiştir. Literatürde bu yönteme isim-fiil (noun-verb) analizi denilmektedir. Abbott'un yaklaşımına göre problemi tanımlayan herhangi bir dokümandaki isimler yazılım sınıflarını, fiiller ise yazılım yordamlarını belirlemede kullanılabilir. Bu yaklaşım daha sonra 1986 yılında Booch [13] tarafından da desteklenerek "nesne yönelimli tasarım yöntemlerinde fiiller yazılım yordamlarını isimler de yazılım sınıflarını belirler" şeklinde tanımlanmıştır. 1987'de Saeki ve arkadaşları da [14] aynı yaklaşımı şu şekilde ifade etmişlerdir: "isimler yazılım sınıfı olarak fiiller ise yazılım yordamı olarak değerlendirilirler".

Elbandak ve arkadaşları [15] kullanım durumu senaryolarındaki isim ve fiilleri belirleyerek UML sınıf diyagramını yarı otomatik olarak oluşturabilmişlerdir. Vidhu Bhala ve Abirami [16], işlevsel tanımlamalardan (functional specifications) yararlanarak isim ve fiilere göre kavramsal model (conceptual model) oluşturmuşlardır. Dennis ve arkadaşları [17] kullanım durumu senaryolarındaki isimlerin olası yazılım sınıflarını işaret ettiğini ve kullanım durumu senaryosundaki fiillerin ise olası yazılım yordamlarını işaret ettiğini vurgulamıştır. Hussain ve arkadaşları [18] yazılım gereksinimlerinden yararlanarak yazılımın işlevsel büyüklüğünü tahmin etmeye çalışmışlardır. Ochodek [19] kullanım durumu isimlerinden veya kullanım durumu diyagramlarından yola çıkarak yazılım büyüklüğünü tahmin etmeye yarayan otomatik bir sistem geliştirmiştir. Heričko ve Živković [20], UML modelleri kullanarak yeni bir yazılım büyüklük kestirimi öne sürmüşleridr. Önerdikleri yöntem Nesne Tabanlı İşlev Puanı yönteminin genişletilmiş versiyonudur.

Birçok araştırmacı nesne yönelimli yazılımların büyüklük ve eforunu tahmin etmek için çalışmalar yürütmüşlerdir. Bu çalışmaların da bazı dezavantajları bulunmaktadır. Örneğin, satır sayısı yöntemi kod yazıldıktan sonra elde edildiği için yazılım büyüklüğünü ölçmek için iyi bir yöntem değildir [21]. İşlev Puanı yönteminde ise elde edilen işlev puanı ölçümleri tek başına bir anlam ifade etmemektedir, ayarlama faktörü kullanılması gerekmektedir [22]. NESMA ve IFPUG yöntemleri ise sadece bilişim sistemleri alanındaki projeler için kullanılmaktadır [22]. COSMIC yöntemi ise sadece bilişim sistemleri alanı, gerçek zamanlı uygulamalar ya da bu ikisinin kombinasyonu olan yazılımların işlevsel büyüklüğünü ölçebilmektedir [7]. Kullanım Durumu Puanı yönteminde kullanılan teknik karmaşıklık faktörü ve çevresel karmaşıklık faktörleri özneldir [23]. Kişiden kişiye değişebilecek bu faktörler yazılımın büyüklüğü ve efor bilgisini doğrudan etkilemektedir [24]. Bütün bu dezavantajlar göz önüne alındığında, tam anlamıyla başarılı ve yeterli bir yöntem bulmak oldukça zordur. Bu nedenle, güvenilir, tutarlı, hızlı ve ucuz maliyetli yazılım büyüklük ve efor kestirim yöntemlerine hala ihtiyaç vardır.

Bu çalışma açık kaynak kodlu yazılımlar üzerinde istatiksel analiz yöntemleri kullanılarak gerçekleştirilmiştir. Açık kaynak kodlu yazılımların kaynak kodlarına erişilebilmesine rağmen çoğunlukla gereksinim belirtimleri yoktur. Bu nedenle, yazılımın özelliklerini tanımlayan gereksinim belirtimleri yerine yazılımın özelliklerini anlatan kullanıcı el kitapları kullanılmıştır. Ayrıca, zaman ve maliyet açısından, çözüm alanı ölçümleri için Understand 2.0 [25] kod analiz yazılımı ve problem tanımı ölçümlerini otomatize etmek için bir Doğal Dil İşleme (Natural Language Processing) aracı olan NLTK kullanılmıştır. NLTK Python programa diliyle geliştirilmiş Doğal Dil İşleme problemlerini gerçekleştirmeye yönelik kullanılan açık kaynak bir araçtır [26]. Bu çalışmadaki asıl amaç kullanıcı el kitaplarından yola çıkarak farklı isim ve farklı fiillerin sayılarını belirleyip, isim sayılarının yazılımdaki sınıf sayılarıyla fiil sayılarının yazılımı oluşturan sınıf ve yordam sayılarıyla ilişkili olup olmadığını incelemektir. Yazılımda tanımlanan sınıf veya yordam sayıları yazılımın büyüklüğünün bir ölçüsü olarak kullanılabilmektedir. Dolayısıyla, eğer problem tanımlarında geçen isim ve fiil sayılarıyla yazılımı oluşturan sınıf ve yordam sayıları arasında bir ilinti varsa, yazılım geliştirilmeden önce yazılım büyüklüğünü tahmin edebilmek mümkün olabilecektir.

Burada dikkat edilmesi gereken en önemli husus, kullanıcı el kitaplarının detay seviyesidir. Herhangi bir belgeleme standardı yoktur. Bir projenin kullanıcı el kitabı çok detaylı yazılabilirken bir başka projenin kullanıcı el kitabı çok sığ yazılabilmektedir. Bu durum analiz sonuçlarını doğrudan etkilemektedir. Bu durumun etkilerini minimize edebilmek için makalenin 2. bölümünde verilen proje seçme kriterleri hassas bir şekilde değerlendirilmeli ve makalenin 4. bölümünde geçerliliğe yönelik tehditler başlığı altında verilen çözüm önerileri dikkate alınmalıdır. Bu makalede önerilen yöntem, diğer yöntemlerin dezavantajları göz önüne alındığında, kullanıcı el kitaplarından otomatik olarak isim ve fiil sayıları belirlenebileceği için hem erken safhada, hem hızlı hem de ucuz bir şekilde yazılımların büyüklük ölçümlerini elde etmeyi sağlamaktadır. Ayrıca bu çalışmanın sonunda elde edilen kazanımlarla, geçmişte yazılımlardaki bir sınıf veya yordamın ortalama ne kadar efor harcanarak geliştirildiğine dair veri toplanmışsa, problem tanımlarındaki isim ve fiil sayıları yazılımın geliştirilmesi için gereken eforun (iş gücünün) tahmini için de girdi olarak kullanılabilir.

Bu makale, daha önce yayınlanan iki [27, 28] çalışmamızın genişletilmesiyle elde edilen sonuçları açıklamaktadır. Önceki çalışmalarımızda sadece isim-sınıf ilişkisi inceleniyorken bu makalede fiil-yordam ilişkisi de incelenmiştir. Daha önce yayınlanan bildirimizde [27] toplam yirmi tane proje incelenmişti. Bu çalışmada ise proje sayısı yirmiyediye çıkarılmıştır. Ayrıca istatiksel analiz ayrıntılandırılmış ve tahmin başarımları Ortalama Bağıl Hata (Mean Magnitude of Relative Error –MMRE), Ortanca Bağıl Hata (Median Magnitude of Relative Error –MdMRE) ve Tahmin Kalitesi (Prediction Quality -Pred (0,25)) kullanarak incelenmiştir [29]. Aykırı değer (outlier analysis) ve normallik (normality analysis) analizleri yanında hata terimi grafikleri (residual plots) de eklenerek istatiksel analizleri derinleştirilmiştir.

Makalede önerilen yazılım büyüklük kestirim yöntemi nicel (quantitative) bir çalışma olduğu için araştırma tasarımı adımları şunlardır:

- Açık kaynak kodlu nesne tabanlı yazılımların problem alanı ölçüleri (isim ve fiil sayıları) ile çözüm alanı ölçüleri (yazılım sınıf ve yordam sayıları) arasındaki ilintinin incelenmesi (correlation analysis),
- Doğrusal bağlanım modelinin bu makaledeki projeler için uygunluğunun incelenmesi,
- Doğrusal bağlanım tabanlı yeni bir nesne tabanlı yazılım büyüklük ölçümü yöntemi önerilmesi,
- 27 farklı açık kaynak kodlu yazılım projeleri ile önerilen yöntemin geçerliliğinin doğrulanması.
- Aşağıda verilen araştırma soruları (AS) 27 açık kaynak kodlu yazılım projeleri yardımı ile cevaplanmıştır:
- AS1: Açık kaynak kodlu nesne tabanlı yazılımların problem alanı ölçüleri (isim ve fiil sayıları) ile çözüm alanı ölçüleri (yazılım sınıf ve yordam sayıları) arasında ilinti (correlation) var mıdır?
- AS2: Bu ilinti yazılım büyüklük ölçümü için kullanılabilir mi?

Makalenin bundan sonraki bölümlerinde bu araştırma sorularına yönelik analizler yer almaktadır. Makalenin 2. bölümünde incelenen projeler tanıtılmaktadır. Makalenin 3. bölümünde isim-sınıf ve fiil-yordam ölçüleri arasındaki ilinti incelenmekte ve bağlanım denklemleri ile elde edilen tahmin başarımları değerlendirilmektedir. 4. bölümde ise yapılan çalışma ile ilgili geçerlilik tehditleri değerlendirilmektedir. Son olarak, makalede elde edilen sonuçlar ve bundan sonra yapılabilecek olası çalışmalar makalenin 5. bölümünde tartışılmaktadır.

2. İNCELENEN PROJELER (ANALYZED PROJECTS)

Bu makalede, isim-sınıf ve fiil-yordam ölçüleri arasındaki ilinti 27 tane açık kaynak kodlu yazılım üzerinde incelenmiştir. İncelenen açık kaynak kodlu yazılım projeleri oyun (14 proje) ve proje yönetimi (13 proje) projeleri olmak üzere iki farklı alandan seçilmiştir. Açık kaynak kodlu yazılımların gereksinim dokümanları olmadığı için bu çalışmada gereksinim dokümanları yerine yazılımın özelliklerini anlatan kullanıcı el kitapları girdi olarak kullanılmıştır.

Projeler Wikipedia ve SourceForge dan belirli kural ve kısıtlar dahilinde seçilmiştir. Bu kısıtlar şunlardır. Projeler açık kaynak kodlu olmalıdır. Yani, projelerin kaynak koduna erişilebilmelidir. Projeler nesne yönelimli bir programlama dili kullanılarak geliştirilmiş olmalıdır. Projelerin kullanıcı el kitapları projelerin resmi internet sayfalarında yayınlanıyor olmalıdır. Burada dikkat edilmesi gereken en önemli husus, kullanıcı el kitaplarının yazılımın kullanımına ilişkin ayrıntılı bilgi veriyor olmasıdır. Kullanılan açık kaynak kodlu projeler ve ilgili web siteleri sırasıyla Tablo 1 ve Tablo 2'de listelenmiştir.

3. İSİM-SINIF VE FİİL-YORDAM ÖLÇÜLERİ ARASINDAKİ İLİNTİ (CORRELATION BETWEEN NOUN-CLASS AND VERB-METHOD MEASURES)

Kullanıcı el kitaplarından isim ve fiil analizi NLTK kullanılarak otomatik olarak gerçekleştirilmiştir. NLTK'nın WordNet Lemmatizer bileşeni kullanılarak otomatik olarak bulunan çoğul isim ve fiiller tekil isim ve fiillere dönüştürülmüştür. Metin içinde birden fazla sayıda geçen isim ve fiiller teke indirilmiştir. Ayrıca eş anlamlı kelimeler dışarıda bırakılmıştır. Son olarakta bulunan isim ve fiiller alfabetik olarak listelenmiş ve belirli bir anlamı olmayan kelimeler sözlüğe başvurularak çıkartılmıştır. Yazılımdaki sınıf ve yordam sayılarını otomatik olarak bulabilmek için de kod analiz etme aracı olan "Understand" [25] kullanılmıştır. Açık kaynak kodlu projeler için ölçü sonuçları Tablo 3'te oyun projeleri için ve Tablo 4'te proje yönetimi projeleri için verilmiştir.

Tablo 1. Oyun projeleri (Game projects)

Proje İsmi	Web Sitesi
Adonthell	http://adonthell.nongnu.org/index.shtml
Exult	http://exult.sourceforge.net/
LinCity	http://lincity.sourceforge.net/
Enigma	http://www.nongnu.org/enigma/
Nuvie	http://nuvie.sourceforge.net/
BattleCity	http://www.battlecity.com.ua/
Rigsof	http://www.rigsofrods.com/content/
BZFlag	http://bzflag.org/
FreeOrion	http://www.freeorion.org/
Wesnoth	http://www.wesnoth.org/
Planeshift	http://www.planeshift.it/
Lierox	http://www.openlierox.net/
CrackAtta	http://www.aluminumangel.org/attack/
Torcs	http://torcs.sourceforge.net/

Tablo 2. Proje yönetimi projeleri (Project management projects)

Proje İsmi	Web Sitesi
LibrePlan	http://www.libreplan.com/
KForge	http://pythonhosted.org/kforge/
GanttProject	http://www.ganttproject.biz/
Tree.io	http://tree.io/
Plandora	http://www.plandora.org/
ProjectLibre	http://www.projectlibre.org/
Project.Net	http://www.project.net/
Scrinch	http://scrinch.sourceforge.net/
Onepoint Project	http://www.onepoint-project.com/home/overview
Task Juggler	http://www.taskjuggler.org/
Sonar Qube	http://www.sonarqube.org/
Freeplane	http://freeplane.sourceforge.net/
OFBiz	http://ofbiz.apache.org/

Tablo 3. Oyun projeleri için isim-sınıf ve fiil-yordam sayıları
(Number of noun-class and verb-method for game projects)

Proje	İsim Sayısı	Fiil Sayısı	Sınıf Sayısı	Yordam Sayısı
Adonthell	84	60	198	1887
Exult*	544	298	595	7432
LinCity	141	87	195	1458
Enigma	462	215	449	6499
Nuvie	229	108	285	5045
BattleCity	81	42	70	848
Rigsof	166	99	257	5356
BZFlag	356	221	747	10531
FreeOrion	336	223	740	14805
Wesnoth	532	305	931	13678
Planeshift	212	106	224	5134
Lierox	288	142	804	14637
CrackAtta	121	88	50	585
Torcs*	722	320	209	4952

***** Bu projeler aykırı değer (outlier) olarak veri kümesinden çıkarılmıştır. Detayları bölüm 3.2'de verilmiştir.

Tablo 4. Proje yönetimi projeleri için isim-sınıf ve fiil-yordam sayıları
(Number of noun-class and verb-method for project management projects)

Proje	İsim Sayısı	Fiil Sayısı	Sınıf Sayısı	Yordam Sayısı
LibrePlan	506	265	3290	23266
KForge	81	48	412	1337
GanttProject	125	52	1300	6954
Tree.io	176	97	618	2474
Plandora	335	145	719	7691
ProjectLibre	537	286	2304	27261
Project.Net	628	352	4058	42953
Scrinch	134	95	286	1495
Onepoint Project	198	104	696	7991
Task Juggler	287	172	332	2323
Sonar Qube	525	236	2970	16643
Freeplane	282	177	2159	12221
OFBiz	355	147	2579	17265

Kullanıcı el kitabındaki farklı isim, farklı fiil sayılarını ve yazılımdaki sınıf ve yordam sayılarını otomatik olarak belirledikten sonra aralarındaki ilinti Pearson İlinti Katsayısı (Pearson's Correlation Coefficient) kullanılarak Minitab istatistik aracıyla hesaplanmıştır. Pearson İlinti Katsayısı –1 ve +1 arasında herhangi bir değer alabilir. Pearson İlinti Katsayısı r_{XY} ile, bağımsız değişken X ile ve tahmin etmeye çalıştığımız bağımlı değişken de Y ile gösterilmektedir. Tablo 5'te tüm açık kaynak kodlu proje alanları için Pearson İlinti Katsayı değerleri verilmiştir.

Tablo 5. Açık kaynak kodlu projelerin pearson ilinti katsayıları
(Pearson correlation coefficients for open source projects)

Projeler	X	Y	r_{XY}	p-değeri
Oyun	İsim	Sınıf	0,834	0,001
	Fiil	Yordam	0,802	0,002
Proje yönetimi	İsim	Sınıf	0,859	0,000
	Fiil	Yordam	0,898	0,000

Tablo 5'te görüldüğü gibi elde edilen tüm Pearson İlinti Katsayıları 0,80 değerinin üzerindedir. Pearson İlinti Katsayısının 0,5 ve 1,0 aralığında olması güçlü ilintiye işaret ettiği için tüm açık kaynak kodlu proje alanları için isim-sınıf ve fiil-yordam ölçüleri arasında yüksek ilinti gözlemlendiği söylenebilmektedir [30]. Ayrıca aralarında yüksek korelasyon olan veri gruplarının sınıflandırma başarısı da artmaktadır [31, 32].

P (Probability; Olasılık) değeri istatistiksel anlamlılığın (statistical significance) varlığının belirlenmesi amacı ile kullanılan bir değerdir. P değerinin 0,05 den küçük olması literatürde "istatistiksel olarak anlamlı" kabul edilmektedir. Tablo 5'te görüldüğü gibi her iki açık kaynak kodlu proje alanları için de p değerleri 0,05 eşik değerinden küçüktür. Dolayısıyla kuracağımız modeller istatiksel olarak anlamlıdır.

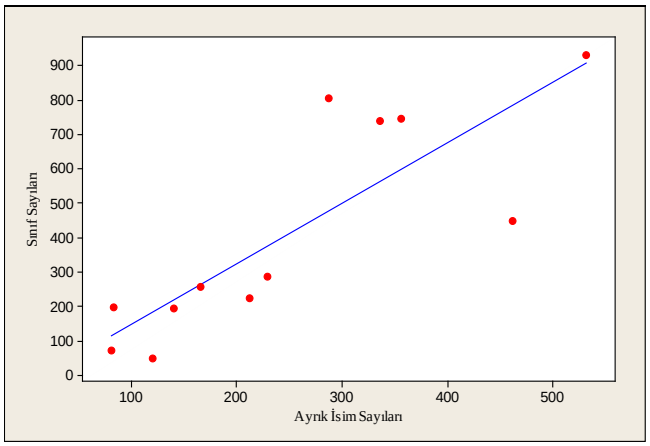
Bağımlı değişkeni tahmin edebilmek için oluşturulacak bağlanım modeli için hataların normal dağılım gösterip göstermediğinin ve Gauss dağılımına uygunluğunun da incelenmesi gerekmektedir. Bu amaçla Ryan Joiner testi kullanılmıştır [33]. Sonuçlar Tablo 6'da verilmiştir.

Tablo 6. Açık kaynak kodlu projeler için ryan-joiner normallik testi sonuçları
(Ryan-joiner normality test for open source projects)

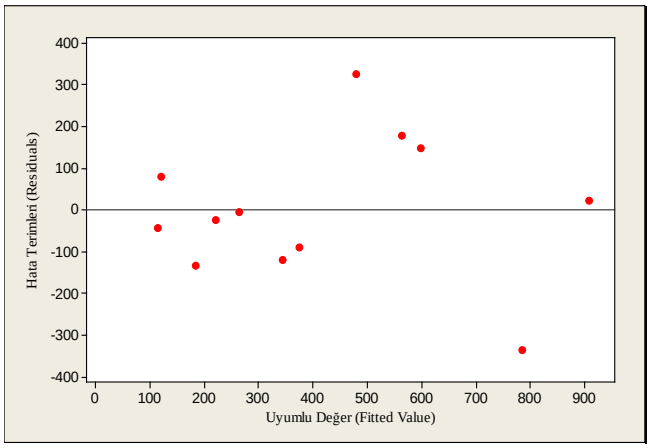
Projeler	İsim ~ Sınıf	Fiil ~ Yordam
Oyun	p-değeri>0,100	p-değeri=0,074
Proje yönetimi	p-değeri>0,100	p-değeri>0,100

Tablo 6'da verilen Ryan-Joiner normallik testi sonuçlarına göre tüm açık kaynak kodlu proje kategorileri için p-değeri 0,05'den büyük çıkmıştır. P-değerinin 0,05'ten büyük olması hataların Gauss dağılımına uygunluğunu göstermektedir. Bu nedenle her iki açık kaynak kodlu proje alanları için bağlanım modeli kurulabileceği açıkça görülmüştür.

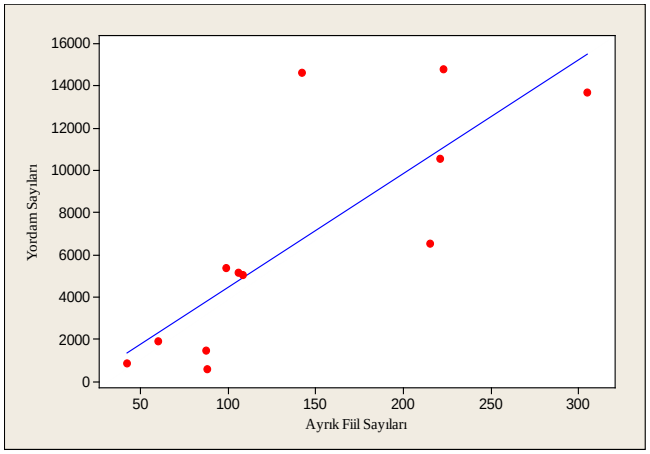
Bağımlı değişkenin gerçek değeri ile bağlanım modeli kullanılarak tahmin edilen değer arasındaki farkı görebilmek için incelenen açık kaynak kodlu projele kategorileri için saçılım grafikleri (scatterplots) ve ilgili hata terimi grafikleri (residual plots) Şekil 1 ve Şekil 8 arasında sırayla verilmiştir. Saçılım grafiği iki değişken arasındaki ilişkinin (ilinti) yönünü, tipini (doğrusal, eğrisel) belirlemeye yarayan bir grafiktir. X ekseninde bağımsız değişken Y ekseninde ise bağımlı değişken verilmektedir.



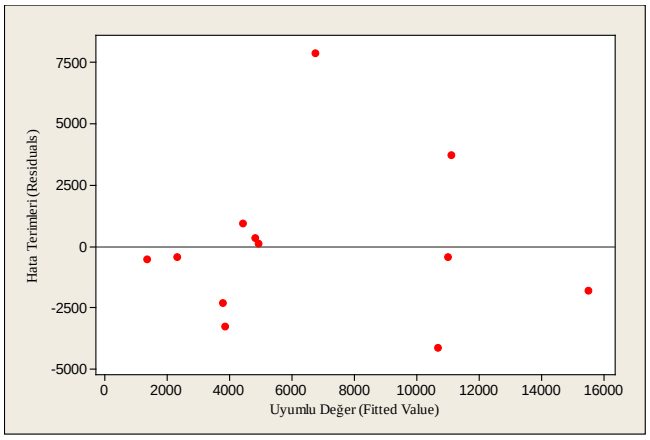
Şekil 1. Oyun projeleri için sınıf sayıları ve ayrık isim sayıları saçılım grafiği
(Scatterplots of number of classes vs. the number of distinct nouns for game projects)



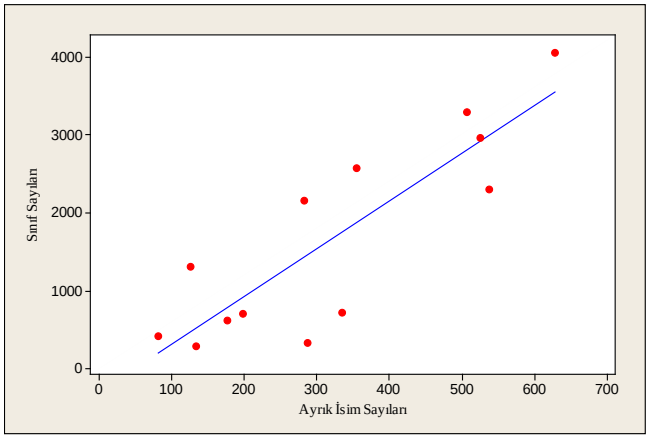
Şekil 2. Oyun projeleri için ayrık isim sayıları ve sınıf sayıları hata terimi grafiği
(The residuals vs. the number of distinct nouns against the number of classes for game projects)



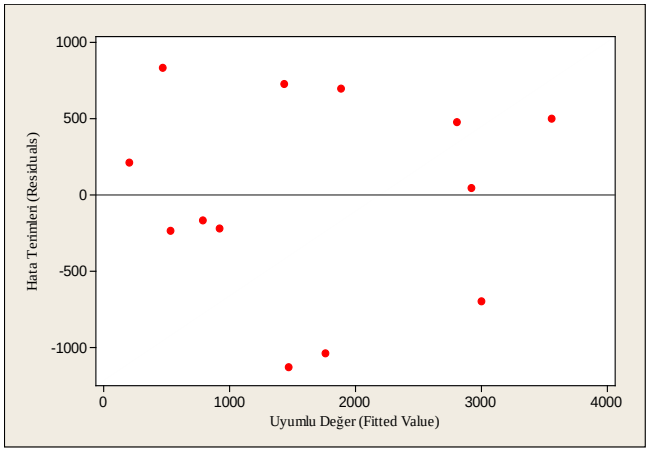
Şekil 3. Oyun projeleri için yordam sayıları ve ayrık fiil sayıları saçılım grafiği
(Scatterplots of number of methods vs. the number of distinct verbs for game projects)



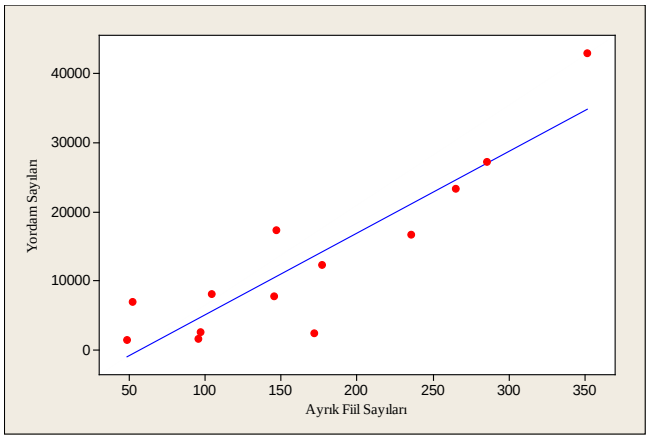
Şekil 4. Oyun projeleri için ayrık fiil sayıları ve yordam sayıları hata terimi grafiği
(The residuals vs. the number of distinct verbs against the number of methods for game projects)



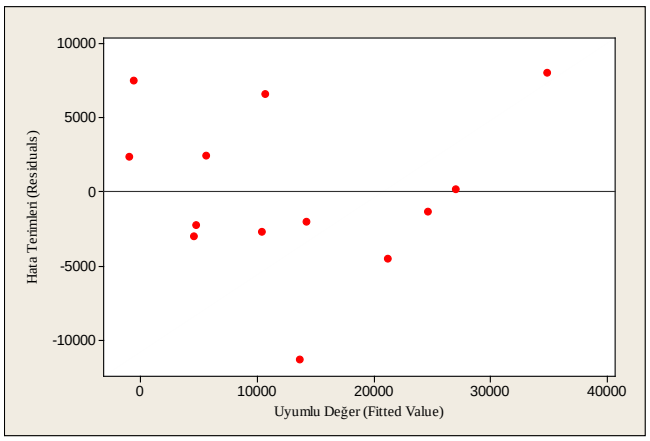
Şekil 5. Proje yönetimi projeleri için sınıf sayıları ve ayrık isim sayıları saçılım grafiği
(Scatterplots of number of classes vs. the number of distinct nouns for project management projects)



Şekil 6. Proje yönetimi projeleri için ayrık isim sayıları ve sınıf sayıları hata terimi grafiği
(The residuals vs. the number of distinct nouns against the number of classes for project management projects)



Şekil 7. Proje yönetimi projeleri için yordam sayıları ve ayrık fiil sayıları saçılım grafiği
(Scatterplots of number of methods vs. the number of distinct verbs for project management projects)



Şekil 8. Proje yönetimi projeleri için ayrık fiil sayıları ve yordam sayıları hata terimi grafiği
(The residuals vs. the number of distinct verbs against the number of methods for project management projects)

Eğer hata terimi grafiğindeki noktalar rastgele dağılmış ve herhangi bir örüntü içermiyolarsa bağımlı değişkeni tahmin edebilmek için doğrusal bağlanım modeli, aksi halde doğrusal olmayan bağlanım modeli daha uygundur [34]. Şekil 2, Şekil 4, Şekil 6 ve Şekil 8'de görüldüğü gibi hata terimi grafiklerinde herhangi bir örüntü görülmemektedir. Değişkenler rasgele dağılım göstermektedir. Bu nedenle, açık kaynak kodlu projelerin sınıf ve yordam sayılarını tahmin etmek için doğrusal bağlanım modelinin uygun olduğu sonucuna varılmıştır.

Doğrusal bağlanım denklemlerinde uygunluk ölçüsü olarak R^2 (belirtme katsayısı) değeri kullanılmaktadır. R^2 bağımlı değişkendeki değişimin % kaçının bağımsız değişkenlerle açıklanabildiğini göstermektedir ve 0 ile 1 arasında bir değer almaktadır. Eğer R^2 değeri, 1'e yakınsa bağımsız değişkenlerin bağımlı değişkeni tam açıkladığı söylenebilir. Öte yandan, R^2 değeri, 0'a yaklaştıkça bağımsız değişkenler bağımlı değişkeni hiç açıklamıyor anlamına gelmektedir. Hastings ve arkadaşlarına göre [35] R^2 0,50 'nin üzerindeyse oluşturulan model güvenilir olarak değerlendirilmektedir.

Oyun projeleri için yazılımın sınıf ve yordam sayısını tahmin etmek için elde edilen doğrusal bağlanım eşitlikleri Eş. 1 ve Eş. 2'de verilmiştir.

$$\text{Sınıf Sayısı} = -28{,}912 + 1{,}761 * \text{İsim Sayısı} \quad (1)$$

$$\text{Yordam Sayısı} = -889{,}4 + 53{,}7 * \text{Fiil Sayısı} \quad (2)$$

Eş. 1 için, R^2=0,700 ve p-value=0,000 olarak hesaplanmıştır. Eş. 2 için ise, R^2=0,644 ve p-value=0,001 hesaplanmıştır. Her iki eşitlik için, R^2 değerleri 0,50'nin üzerinde olduğu için oluşturulan modeller güvenilir olarak değerlendirilebilir. P-value değerleride 0,05 eşiğinin altında olması da kurulan modellerin istatistiksel olarak anlamlı olduğunu göstermektedir.

Proje Yönetimi projeleri için yazılımın sınıf ve yordam sayısını tahmin etmek için elde edilen doğrusal bağlanım eşitlikleri Eş. 3 ve Eş. 4'te verilmiştir.

$$\text{Sınıf Sayısı} = -300{,}35 + 6{,}15 * \text{İsim Sayısı} \quad (3)$$

$$\text{Yordam Sayısı} = -6710{,}4 + 118{,}2 * \text{Fiil Sayısı} \quad (4)$$

Eş. 3 için, R^2=0,738 ve p-value=0,000 olarak bulunmuştur. Eş. 4 için ise, R^2=0,807 ve p-value=0,000 olarak hesaplanmıştır. Her iki eşitlik için, R^2 değerleri 0,50'nin üzerinde olduğu için oluşturulan modeller güvenilir olarak değerlendirilmiştir. P-value değerlerinin 0,05 eşiğinin altında olması da kurulan modellerin istatistiksel olarak anlamlı olduğunu göstermektedir.

3.1. Doğruluk Tahmini (Prediction Accuracy)

Bağıl Hata (Magnitude of Relative Error - MRE), Ortalama Bağıl Hata (Mean Magnitude of Relative Error - MMRE), Ortanca Bağıl Hata (Median Magnitude of Relative Error - MdMRE), Tahmin Kalitesi (Prediction Quality -Pred(e)) ve Ortalama Karesel Hata (Mean Square Error - MSE) kestirim doğruluğu için en sık kullanılan ölçülerdir. MRE tekil kestirim hata oranıdır, MMRE bir veri kümesinde göreceli hataların yüzde olarak ortalamasıdır ve MdMRE ise bir veri kümesinde göreceli hataların yüzde olarak ortancasıdır. Pred(e), kestirimlerin yüzde kaçının % e hata oranı içinde olduğunu göstermektedir. Pred değerleri hesaplanırken "e" sayısı hem 0,25 hem de 0,30 olarak alınmıştır çünkü bu sayılar endüstride kestirim modelleri yaratırken en sık kullanılan sayılardır. MSE ise toplam karesel hatanın proje sayısına bölünmesiyle elde edilmiştir. En iyi MSE değeri en düşük MSE değeridir.

En iyi MMRE ve MdMRE en düşük MMRE ve MdMRE değerleridir çünkü hata oranının en düşük seviyede olması kestirimler için en iyi sonucu vermektedir. Diğer yandan en iyi pred(0,25) ve pred(0,30) değeri en yüksek değerdir çünkü bu değer geliştirilen matematiksel modellerin doğruluğunu göstermektedir. MMRE ve MdMRE nin birbirinden en önemli farkı, MMRE'nin büyük MRE değerlerine karşı daha hassas olmasıdır. Conte, Dunsmore ve Shen'e [36] göre iyi

bir kestirim için MMRE ve MdMRE değerlerinin 0,25'ten az olması gerekmektedir. Diğer taraftan Hastings ve Sajeev' e [35] göre de MMRE ve MdMRE değerleri 0,20'den az ise öngörücü (predictive), 0,20 ile 0,50 arasında ise kabul edilebilir (acceptable), ve 0,50'den büyük ise kabul edilemez (unacceptable) olarak değerlendirilmektedir. Ancak MRE tabanlı doğruluk ölçümleri yazılım mühendisliği alanındaki birçok araştırmacı tarafından eleştirilmektedir [37, 38]. Foss ve arkadaşları [37] MMRE'nin güvenilir bir doğruluk ölçümü olup olmadığını ortaya koymak için bir simulasyon çalışması yapmışlardır. Bu çalışmanın sonucunda MMRE 'nin birçok durum için güvenilmez olduğu ortaya çıkmıştır. Shepperd ve MacDonell de [38] yaptıkları çalışmada M(MRE) kullanımının doğru sonuç vermeyeceğini belirtmişlerdir. Jørgensen de [39] yaptığı çalışmada MMRE'nin çok büyük MRE değerlerine karşı daha hassas olmasından dolayı MMRE yerine MdMRE'yi kullanılmayı tavsiye etmiştir.

MRE, MMRE, MdMRE, Pred(e) ve MSE eşitlikleri Eş. 5, Eş. 6, Eş. 7, Eş. 8 ve Eş. 9'da verilmiştir.

$$MRE = \frac{\left|E_{tahmin} - E_{gerçek}\right|}{E_{gerçek}} \quad (5)$$

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|E_{tahmin} - E_{gerçek}\right|}{E_{gerçek}} \quad (6)$$

$$\text{MdMRE} = \text{medyan}\ (\text{MRE}_i) \quad (7)$$

$$\text{Pred(e)} = k/n \quad (8)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(E_{tahmin} - E_{gerçek})^2 \quad (9)$$

Eş. 8'deki "k", MRE'si "e" değerinden daha düşük olan proje sayısını temsil etmektedir. "e" değeri MRE değerleri için seçilen eşik değeridir. Eşitliklerdeki "n" ise veri kümesindeki toplam proje sayısını temsil etmektedir. Conte, Dunsmore ve Shen'e göre [40] Pred(0,25) değeri 0,75'e eşit veya daha yüksek olmalıdır [36]. Diğer taraftan, Tate ve Verner daha gerçekçi bir kestirim modeli için Pred(0,30) kullanılmalı ve Pred(0,30) değeri 0,70'i geçmelidir demişlerdir. Ayrıca Kitchenham ve arkadaşları [41], pred(e)'nin eşik değerinin üstündeki değerler için hassas olmadığını belirtmişlerdir. Örneğin, pred (0,25) değerinin MRE'si 0,26 olan bir projeyle MRE'si 2,60 olan bir projeyi ayırt edemeyeceğini söylemişlerdir.

Tablo 7 ve Tablo 8'de açık kaynak kodlu projeler için doğruluk tahminleri (accuracy prediction) verilmiştir. Tablo 7'deki tahminler öncelikle tüm projeler için sonrasında da her bir adımda çıkarılan aykırı değerlere göre verilmiştir. Bu tablolarda, Y~X gösteriminde Y tahmin etmeye çalıştığımız bağımlı değişkeni X ise bağımsız değişkeni temsil etmektedir. Elde edilen sonuçlara göre tüm projelerin birlikte değerlendirildiği sonuçlarla tüm aykırı değerlerin çıkarılması sonucu elde edilen sonuçlar arasında tüm doğruluk tahmin ölçüleri için (MMRE, MdMRE, pred (0,25) ve pred(0,30)) genel bir iyileşme görülmektedir.

Tablo 7. Oyun projeleri için doğruluk tahminleri (Prediction accuracy for game projects)

Proje	Tüm Projeler		Torcs Projesi Çıkarıldığında		Torcs ve Exult Projeleri Çıkarıldığında	
	İsim~Sınıf	Fiil~Yordam	İsim~Sınıf	Fiil~Yordam	İsim~Sınıf	Fiil~Yordam
	MRE	MRE	MRE	MRE	MRE	MRE
Adonthell	0,236	0,806	0,255	0,506	0,399	0,237
Exult*	0,009	0,438	0,391	0,724	-	-
LinCity	0,474	1,904	0,188	1,725	0,125	1,596
Enigma	0,176	0,254	0,574	0,436	0,747	0,640
Nuvie	0,240	0,033	0,269	0,038	0,313	0,026
BattleCity	2,466	2,370	1,041	1,462	0,624	0,612
Rigsof	0,191	0,140	0,045	0,164	0,024	0,172
BZFlag	0,399	0,208	0,263	0,089	0,199	0,043
FreeOrion	0,413	0,432	0,296	0,346	0,239	0,250
Wesnoth	0,376	0,202	0,129	0,041	0,024	0,133
Planeshift	0,521	0,061	0,503	0,071	0,537	0,063
Lierox	0,505	0,595	0,441	0,571	0,405	0,539
CrackAtta	4,452	6,291	3,041	5,863	2,683	5,563
Torcs*	2,458	1,295	-	-	-	-
Doğruluk Tahminleri	pred(0,25)=0,35 pred(0,30)=0,35 MMRE =0,92 MdMRE=0,40 MSE=6085	pred(0,25)=0,35 pred(0,30)=0,42 MMRE =1,07 MdMRE=0,43 MSE=1517047	pred(0,25)=0,23 pred(0,30)=0,42 MMRE =0,57 MdMRE=0,29 MSE=3136	pred(0,25)=0,35 pred(0,30)=035 MMRE =0,92 MdMRE=0,43 MSE=1201509	pred(0,25)=0,41 pred(0,30)=0,41 MMRE =0,52 MdMRE=0,35 MSE=2684	pred(0,25)=0,50 pred(0,30)=0,58 MMRE =0,82 MdMRE=0,24 MSE=953768

Tablo 8. Proje yönetimi projeleri için doğruluk tahminleri
(Prediction accuracy for project management projects)

Proje	İsim ~ Sınıf	Fiil ~ Yordam
	MRE	MRE
LibrePlan	0,145	0,057
KForge	0,520	1,776
GanttProject	0,640	1,081
Tree.io	0,264	0,920
Plandora	1,446	0,355
ProjectLibre	0,302	0,006
Project.Net	1,122	0,187
Scrinch	0,830	2,019
Onepoint Project	0,317	0,301
Task Juggler	3,409	4,859
Sonar Qube	0,014	0,272
Freeplane	0,336	0,162
OFBiz	0,270	0,382
Doğruluk Tahminleri	pred(0,25)=0,231 pred(0,30)=0,385 MMRE =0,663 MdMRE=0,317	pred(0,25)=0,308 pred(0,30)=0,385 MMRE = 0,953 MdMRE=0,355

Bu da yapılan aykırı değer analizinin sonuçları olumlu anlamda etkilediğini gözler önüne sermektedir. Doğruluk tahmini için kullanılan ölçülere tek tek bakıp farklı araştırmacılar tarafından farklı belirlenen eşik değerlerini yorumlamak yerine tüm modeller için hesaplanan MSE değerlerine bakarak birbirleriyle kıyaslamak daha anlamlıdır. MSE değeri ne kadar küçükse hata miktarı da o kadar azdır. Tüm projeler için elde edilen sonuçlara göre hem İsim~Sınıf modelinin hem de Fiil~Yordam modelinin MSE değerleri aykırı değerlerin çıkarılmasıyla elde edilen MSE değerlerine göre oldukça yüksektir. Her bir aykırı değerin modelden çıkarılmasıyla MSE değerleri düşmektedir.

En düşük MSE değerlerinin tüm aykırı değerlerin atıldığı modellerden elde edildiği açıkça görülmektedir. Tüm aykırı değerlerin atıldığı modeller için doğruluk tahmini sonuçlarını değerlendirmek istersek, hem İsim~Sınıf hem de Fiil~Yordam modeli için kabul edilebilir MdMRE sonuçları gözlemlenmiştir [35]. Hem İsim~Sınıf hem de Fiil~Yordam modelleri için MMRE değerleri eşik değerinin üstündedir. Pred(0,25) ve pred(0,30) değerleri ise eşik değerinin altındadır. Fakat İsim~Sınıf modeli için 0,313 MRE değerine sahip bir proje bulunmaktadır. Bu değer pred(0,30) eşik değerinin çok az üzerindedir. Tablo 8'de proje yönetimi projeleri için verilen doğruluk tahmini sonuçlarına göre hem İsim~Sınıf hem de Fiil~Yordam modeli için kabul edilebilir MdMRE sonuçları gözlemlenmiştir [35].

Hem İsim~Sınıf hem de Fiil~Yordam modelleri için MMRE, değerleri eşik değerinin üstündedir. Pred (0,25) ve pred(0,30) değerleri ise eşik değerinin altındadır. Fakat üç projenin MRE değerleri İsim~Sınıf modeli için 0,302, 0,317 ve 0,336'dır. Bu değerler pred(0,30) eşik değerinin çok az üstündedir. Bu nedenle İsim~Sınıf modeli için pred (0,30) değerleri kabul edilebilir olarak değerlendirilebilir. Ayrıca Fiil~Yordam modeli için de bir projenin MRE değeri 0,301'dir. Bu değer de pred(0,30) eşik değerinin çok az üzerindedir.

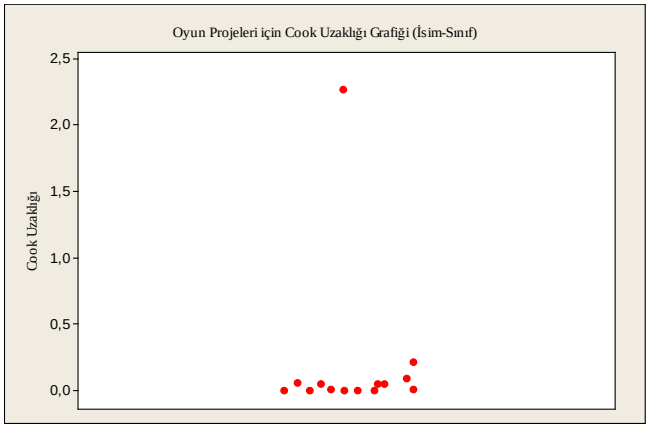
3.2. Aykırı Değer Analizi (Outlier Analysis)

Rousseeuw ve Zomeren'a göre [42] aykırı gözlemler, verideki toplam gözlem sayısının yarısından daha az sayıda olmasına rağmen, o verideki gözlemlerin çoğunun vermek istediği bilgiye engel olan ve sonuçlar üzerinde yanıltıcı bir etki yaratabilen gözlemlerdir. Veri kümesinde bulunan aykırı gözlemler yapacağımız istatiksel analizlerin etkilenmesine sebep olabilmektedir. Bu nedenle veri kümesindeki aykırı değerler her iki açık kaynak kodlu proje alanları için belirlenmiştir. Bir veri kümesindeki değerin aykırı değer olup olmadığını belirlemek için geliştirilmiş birçok teknik bulunmaktadır. Acarlar ve Gamgam [43], etkili gözlem grubu içeren veri kümeleri için, Cook Uzaklığı (Cook's Distances) ve diğer teknikleri karşılaştırmış ve Cook Uzaklığı tekniğinin aykırı değerleri daha iyi saptadığı sonucuna varmışlardır. Ayrıca Cook Uzaklığı tekniği regresyon analizinde aykırı değerlerin saptanmasında, uygulama kolaylığından dolayı diğer tekniklere kıyasla daha kullanışlı olduğu için bu çalışmada tercih edilmiştir. Bu teknikte veri kümesindeki projelerin Cook uzaklığı 4/n (n:toplam proje sayısı) eşik değerinden büyükse o proje aykırı değer olarak değerlendirilmektedir [44]. Bu çalışmada, herhangi bir projeye aykırı değer diyebilmek için o projenin Cook uzaklığının hem İsim~Sınıf hem de Fiil~Yordam modelleri için 4/n eşik değerinden yüksek olması gerekmektedir. Tek bir model için bu analiz yapıldığında çok fazla aykırı değer veri kümesinden çıkarılması gerekmektedir. Zaten veri kümesinin sayıca az olduğu düşünülürse geriye kalan projelerle istatiksel analiz yapmak daha da zorlaşacaktır. Tablo 9'da oyun projeleri için aykırı değer analizi sonuçları verilmiştir.

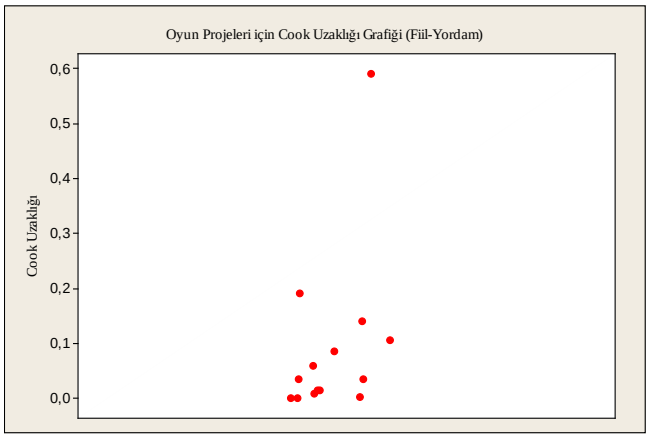
Tablo 9. Oyun projeleri için aykırı değer analizi
(Outlier analysis for game projects)

Proje	İsim ~ Sınıf	Fiil ~ Yordam
	Cook Uzaklığı	Cook Uzaklığı
Adonthell	0,00382	0,015326
Exult	0,00005	0,107238
LinCity	0,00991	0,034662
Enigma	0,00689	0,008631
Nuvie	0,00329	0,000095
BattleCity	0,05246	0,035142
Rigsof	0,00241	0,002175
BZFlag	0,05612	0,016229
FreeOrion	0,05628	0,141688
Wesnoth	0,21890	0,086400
Planeshift	0,01031	0,000353
Lierox	0,09706	0,190956
CrackAtta	0,06545	0,060061
Torcs	2,26641	0,591949

Bu sonuçlara göre Cook uzaklığı eşik değeri azami 4/14=0,285 olmalıdır. Fakat Torcs projesinin Cook uzaklığı hem İsim~Sınıf hem de Fiil~Yordam modelleri için bu eşik değerinden yüksektir. Böylece Torcs projesi aykırı değer olarak değerlendirilip veri kümesinden çıkarılmıştır. 14. proje olan Torcs projesinin hem isim-sınıf hem de fiil-yordam sayıları için cook uzaklığı grafikleri sırasıyla Şekil 9 ve Şekil 10'da verilmiştir.



Şekil 9. Oyun projeleri için isim-sınıf analizinden elde edilen cook uzaklığı grafiği
(Cook distance graph according to noun-class analysis for game projects)



Şekil 10. Oyun projeleri için fiil-yordam analizinden elde edilen cook uzaklığı grafiği
(Cook distance graph according to verb-method analysis for game projects)

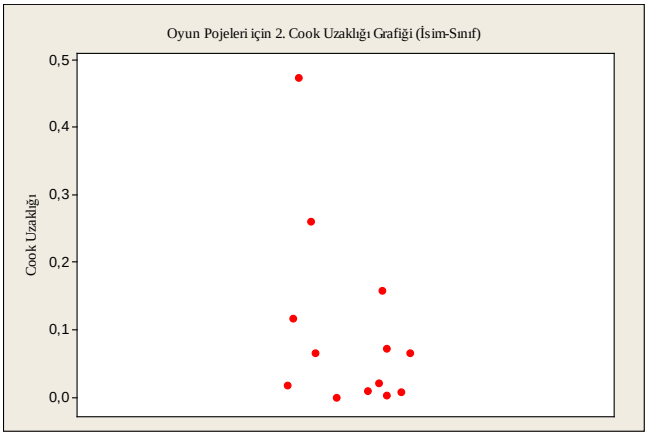
Torcs projesinin veri kümesinden çıkarılmasından sonra tekrar yapılan aykırı değer analiz sonuçları Tablo 10'da verilmiştir. Yeni eşik değerin 4/13=0,307 olduğu unutulmamalıdır.

Tablo 10. Torcs projesinin çıkarımından sonraki aykırı değer analizi (Outlier analysis after removal of torcs project)

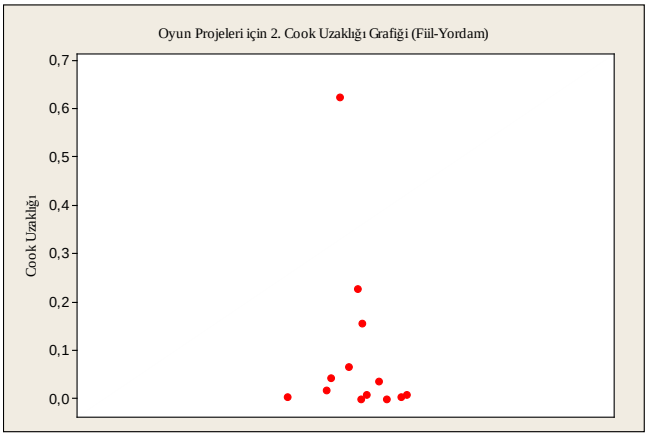
Proje	İsim ~ Sınıf	Fiil ~ Yordam
	Cook Uzaklığı	Cook Uzaklığı
Adonthell	0,010261	0,007937
Exult	0,472944	0,623205
LinCity	0,003187	0,036026
Enigma	0,259962	0,042961
Nuvie	0,007911	0,000151
BattleCity	0,021666	0,018085
Rigsof	0,000251	0,003684
BZFlag	0,065612	0,005043
FreeOrion	0,072314	0,156361
Wesnoth	0,117216	0,007623
Planeshift	0,018507	0,000570
Lierox	0,158883	0,226815
CrackAtta	0,065841	0,065984

Exult projesini Cook uzaklığı hem İsim~Sınıf hem de Fiil~Yordam modelleri için bu eşik değerinden yüksektir. Böylece Exult projesi de aykırı değer olarak değerlendirilip veri kümesinden çıkarılmıştır.

2. proje olan Exult projesinin hem isim-sınıf hem de fiil-yordam sayıları için cook uzaklığı grafikleri sırasıyla Şekil 11 ve Şekil 12'de verilmiştir.



Şekil 11. Oyun projeleri için isim-sınıf analizinden elde edilen 2. cook uzaklığı grafiği
(2nd cook distance graph according to noun-class analysis for game projects)



Şekil 12. Oyun projeleri için fiil-yordam analizinden elde edilen 2. cook uzaklığı grafiği
(2nd cook distance graph according to verb-method analysis for game projects)

Exult projesinin veri kümesinden çıkarılmasından sonra tekrar yapılan aykırı değer analiz sonuçları Tablo 11'de verilmiştir. Yeni eşik değeri 4/12=0,333 olmaktadır ve Cook uzaklığı bu eşik değerini Enigma projesi için İsim~Sınıf modeli için aşmıştır. Ancak Fiil~Yordam modeli için bu eşik değeri aşılmamıştır. Projenin aykırı değer olabilmesi için her iki model için de eşik değerini aşması gerektiği için Enigma projesi aykırı değer olarak değerlendirilmemiştir. Eşik değerini geçen başka bir proje bulunmadığından oyun projeleri için aykırı değer analizi burada tamamlanmıştır. 3. proje olan Enigma projesinin hem isim-sınıf hem de fiil-yordam sayıları için cook uzaklığı grafikleri sırasıyla Şekil 13 ve Şekil 14'te verilmiştir.

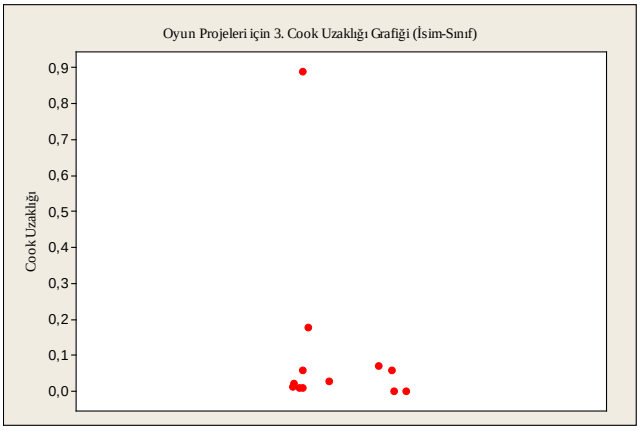
Tablo 11. Exult projesinin çıkarımından sonraki aykırı değer analizi (Outlier analysis after removal of exult project)

Proje	İsim ~ Sınıf	Fiil ~ Yordam
	Cook Uzaklığı	Cook Uzaklığı
Adonthell	0,030258	0,002263
LinCity	0,001646	0,038455
Enigma	0,890921	0,170442
Nuvie	0,012631	0,000091
BattleCity	0,009568	0,004294
Rigsof	0,000092	0,005100
BZFlag	0,059141	0,002269
FreeOrion	0,070723	0,156487
Wesnoth	0,010071	0,224970
Planeshift	0,024327	0,000584
Lierox	0,177044	0,270139
CrackAtta	0,060073	0,074011

Tablo 12'de proje yönetimi projeleri için aykırı değer analizi sonuçları verilmiştir.

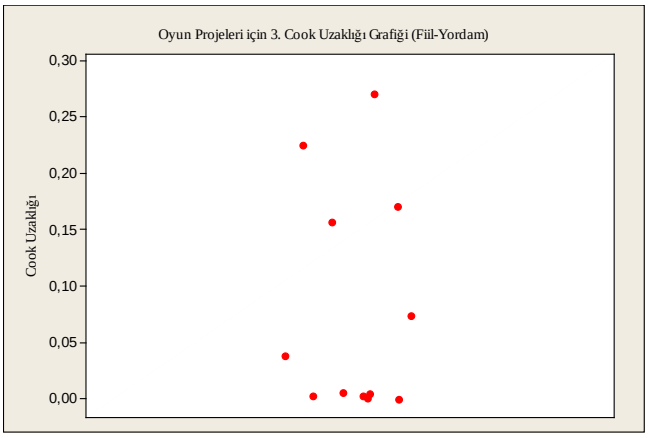
Tablo 12. Proje yönetimi projeleri için aykırı değer analizi (Outlier analysis for project management projects)

Proje	İsim ~ Sınıf	Fiil ~ Yordam
	Cook Uzaklığı	Cook Uzaklığı
LibrePlan	0,057738	0,00673
KForge	0,018184	0,03017
GanttProject	0,188824	0,28315
Tree.io	0,004893	0,01308
Plandora	0,103878	0,01128
ProjectLibre	0,157155	0,00017
Project.Net	0,181510	1,13628
Scrinch	0,014286	0,02365
Onepoint Project	0,007615	0,01336
Task Juggler	0,127530	0,18017
Sonar Qube	0,000544	0,05056
Freeplane	0,053165	0,00561
OFBiz	0,048449	0,06522



Şekil 13. Oyun projeleri için isim-sınıf analizinden elde edilen 3. cook uzaklığı grafiği
(3rd cook distance graph according to noun-class analysis for game projects)

Cook uzaklığı tekniği için eşik değeri 4/13=0,307'dir ve Cook uzaklığı bu eşik değerini hem İsim~Sınıf hem de Fiil~Yordam modelleri için geçen başka bir proje bulunmamaktadır. Bu nedenle proje yönetimi projeleri için aykırı değer analizi burada tamamlanmıştır.



Şekil 14. Oyun projeleri için fiil-yordam analizinden elde edilen 3. cook uzaklığı grafiği
(3rd cook distance graph according to verb-method analysis for game projects)

4. GEÇERLİLİĞE YÖNELİK TEHDİTLER (THREATS TO VALIDITY)

Deneysel çalışmaların bir sonucu olarak bazı geçerlilik tehditleri oluşlabilmektedir. Veri kümesinin sınırlı sayıda olması geçerliliği tehdit eden unsurlardan biridir. Bu çalışmada 27 adet açık kaynak kodlu yazılım incelenmiştir. Ancak, bu çalışmada kullanılan veri kümesi daha önce yapılan bazı çalışmalarda kullanılan veri kümelerinden hala daha fazladır [24, 45].

Ne var ki veri kümesindeki proje sayısını artırmadan sonuçların genellenebilir olduğuyla ilgili herhangi bir şey söylenememektedir. Açık kaynak kodlu yazılımlar için hangi projelerin seçildiği de üzerinde durulması gereken bir geçerlilik tehdidir. Bu tehditi ortadan kaldırmak için makalenin 2. bölümünde verilen proje seçim kriterleri hassas olarak uygulanmış bu kriterlere uymayan hiçbir proje veri kümesine dahil edilmemiştir. Ayrıca önerilen yöntemin doğruluğunu tespit edebilmek için projeler tek bir proje alanından değil 2 farklı alandan seçilmiştir.

Diğer bir geçerlilik tehditi ise isim ve fiil sayılarının belirlenmesi esnasında yapılabilecek hatalar ve yaşanabilecek öznelliktir (subjectivity). Bu tehditin önüne geçebilmek için isim ve fiiller NLTK aracı kullanılarak otomatik olarak elde edilmiştir. Aynı projeler için NLTK aracı ile kişilerden bağımsız olarak aynı sonuçlar elde edilecektir. Aynı şekilde yazılım sınıflarının ve yazılım yordamlarının kişiden bağımsız belirlenmesinde Understand otomatik kod analiz aracı kullanılmıştır. Bu araç benzer çalışmalarda nesnellik için tercih edilen olgun bir araçtır [46]. Bu çalışmada temel alınan kullanıcı el kitabı detay seviyesi elde edilen sonuçları doğrudan etkileyen ve geçerliliği tehdit eden bir unsurdur. Ancak, bir projenin

kullanıcı el kitabı detay seviyesini değerlendirebilmek için, proje kullanıcılarının yorumu gereklidir. Öznel bir değerlendirme yerine, bölüm 3.2'de anlatıldığı gibi, ele alınan bir projeyi diğer projelerle birlikte değerlendirerek Cook uzaklığı yöntemine göre ayrıştırmak daha nesnel bir sonuç alınmasını sağlayacaktır. Yeni bir proje üzerinde çalışmak isteyen araştırmacılar, değerlendirecekleri projeyi bu çalışmada ele alınan proje alanlarından uygun olan proje kümesine dahil ederek Cook uzaklığı analizi yapmak suretiyle, projenin bu çalışmada anlatılan büyüklük kestirim yöntemine uygunluğunu belirleyebilirler. Bu uygunluk, kullanıcı el kitabı detay seviyesinin yeterliliği olarak da değerlendirilebilir.

5. SONUÇLAR VE TARTIŞMALAR (RESULTS AND DISCUSSIONS)

Bu çalışmada açık kaynak kodlu nesne yönelimli projelerin kullanıcı el kitaplarındaki farklı isim ve farklı fiil sayıları ile yazılımdaki sınıf ve yordam sayıları otomatik olarak belirlenerek aralarındaki ilinti incelenmiştir. Gözlemlenen ilintiden yola çıkarak projenin başlangıç aşamalarında kullanıcı el kitaplarındaki farklı isim ve fiil sayılarına bakarak yazılımın sınıf ve yordam sayılari ile ilgili fikir sahibi olunabileceği gösterilmiştir. Genel olarak yazılım şirketleri ortalama olarak bir yazılımdaki sınıfın ya da yordamın ne kadar sürede yazılabileceğini geçmiş tecrübelerinden bilmektedirler. Dolayısıyla tasarıma veya kodlamaya geçmeden gereksinimlerde geçen farklı isim ve fiil sayılarını belirlemek büyüklük tahmini yanında efor tahmini için de kullanılabilir. Bu çalışmada, 27 adet açık kaynak kodlu yazılım incelenmiştir. Yapılan ilinti analizi sonucuna göre isim-sınıf ve fiil-yordam sayıları arasında güçlü bir ilinti gözlemlenmiştir. Bu ilintidan faydalanarak projelerin büyüklüğü ile ilgili kestirim yapılabilir mi sorusuna cevap verebilmek için de birtakım istatiksel analizler yapılmıştır. Bu analizlere göre incelenen projelerin birçoğu için kabul edilebilir pred (0,25), pred (0,30), MMRE ve MdMRE değerleri elde edilebilmiştir. Yapılan çalışmanın bazı kısıtları vardır.

Elde edilen bağlanım denklemlerinde eksi sabitler bulunmaktadır ve bu denklemler isim ve fiil sayıları küçük olduğu zaman geçersiz sonuçlar verebilmektedir. Bu nedenle bu türden denklemler kullanılırken dikkat etmek gerekmektedir. Öte yandan, açık kaynak kodlu projeleri kullanmanın bazı dezavantajları vardır. Mesela; her proje farklı kişiler tarafından gerçekleştirilmiş ve kullanıcı el kitapları hazırlanmıştır. Bu durum projelerin birlikte değerlendirilmesi konusunda sorun yaratmaktadır. Herhangi bir belgeleme standardı yoktur. Bir projenin kullanıcı el kitabı çok detaylı yazılabilirken bir başka projenin kullanıcı el kitabı çok sığ yazılabilmektedir. Bu nedenlerle, yapılan analizlerde çok yüksek kestirim başarımı beklemek doğru olmaz. Gelecekte doğrudan yazılım gereksinim dökümanını (Software Requirement Sprecification-SRS) girdi olarak alıp yazılımı üretecek (kaynak kod) araç geliştirilebilir. Yapılan bu çalışmanın yazılım gereksinim dökümanı ile hedef yazılım arasındaki ilişkinin kurulmasında önemli bir rol oynayacağı düşünülmektedir. Nitekim bu çalışma yazılım gereksinim dökümanındaki isimler ile sınıfların, fiiller ile de yordamların arasında doğrudan bir ilişki olduğu esasına dayanmakta ve isim ve fiillerin nasıl ayrıştırılacağını ortaya koymaktadır. Bu konuda çalışma yapacak araştırmacıların böyle bir gelişmeyi de hedefleyerek yazılım geliştirme yönetimi alanında önemli katkılar sağlayacağı düşünülmektedir. Bu çalışmanın devamında incelenen proje sayısının artırılması ve sonuçların geçerliliğinin incelenmesi planlanmaktadır. Yazılım büyüklüğü ile ilgili olarak yazılım için harcanan zamanında değerlendirmeye katılması düşünülmektedir. Ayrıca benzer bir çalışmanın kontrat kapsamında veya ticari amaçla geliştirilen yazılımlarla da yapılması ve problem alanı büyüklüğü ile yazılım büyüklüğünün ilişkilendirilmesinin yanısıra geliştirme için gereken eforun da ilişkilendirilmesi bu çalışmada elde edilen sonuçların kullanılabilirliği açısından önem arz etmektedir.

KAYNAKLAR (REFERENCES)

1. Živković A., Rozman I., Herićko M., Automated software size estimation based on function points using UML models, Information and Software Technology, 47 (13), 881-890, 2005.
2. Azzeh M., Nassif A.B., Fuzzy Model Tree for Early Effort Estimation, 12th International Conference on Machine Learning and Applications (ICMLA), Florida-USA, 117-121, December 4-7, 2013.
3. Nassif A.B., Ho D., Capretz L.F., Towards an early software estimation using log-linear regression and a multilayer perceptron model, Journal of Systems and Software, 86 (1), 144-160, 2013.
4. Albrecht A., Measuring Application Development Productivity, In Proceedings of the Joint SHARE/GUIDE, and IBM Application Development Symposium, Monterey-California, 83-92, October 14-17, 1979.
5. ISO/IEC, ISO/IEC 20926: Software a nd Systems Engineering — Software Measurement — IFPUG Functional Size Measurement Method, International Organization for Standardization, Geneva, Switzerland, 2009.
6. ISO/IEC, ISO/IEC 20968: Software Engineering Mk II Function Point Analysis — Counting Practices Manual, International Standardization Organization, Geneva, Switzerland, 2002.
7. COSMIC, Measurement Manual, Version 3.0.1, Common Software Measurement International, 2009.
8. Banker R., Kauffman R., Wright C., Zweig D., Automating Output Size and Reuse Metrics in a Repository-Based Computer Aided Software Engineering (CASE) Environment, IEEE Transaction on Software Engineering, 20 (3), 169-186, 1994.
9. Antoniol G., Fiutem R., Lokan, C., Object-Oriented Function Points: An Empirical Validation, Empirical Software Engineering, 8 (3), 225-254, 2003.
10. Karner, G., Metrics for objector, Diploma thesis, University of Linkoping, Sweden,1993.
11. Costagliola G., Ferrucci F., Tortora G., Vitiello G., A metric for the size estimation of object oriented

graphical user interfaces, International Journal of Software Engineering and Knowledge Engineering, 10 (5), 581–603, 2000.

12. Abbott R.J., Program design by informal English descriptions, Communications of the ACM, 26, 882–894, 1983.

13. Booch G., Object-oriented development, IEEE Transactions on Software Engineering, 12 (2), 211–221, 1986.

14. Saeki M., Horai H., Toyama K., Uematsu N., Enomoto H., Specification framework based on natural language, In Proceedings of the 4th international Workshop on Software Specification and Design, Monterey-CA, 87–94, April 3-4, 1987.

15. Elbendak M., Vickers P., Rossite N., Parsed use case descriptions as a basis for object-oriented class model generation, Journal of Systems and Software, 84, 1209-1223, 2011.

16. Vidhu Bhala R.V., Abirami S., Conceptual modeling of natural language functional requirements, Journal of Systems and Software, 88, 25-41, 2014.

17. Dennis A., Wix B.H., Tegarden F., Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 2nd edition, John Wiley & Sons, New York, A.B.D., 199-201, 2009.

18. Hussain I., Kosseim L., Ormandjieva O., Approximation of COSMIC functional size to support early effort estimation in Agile, Data&Knowledge Engineering, 85, 2-14, 2013.

19. Ochodek M., Functional Size Approximation based on use case names, Information and Software Technology, 80, 73-88, 2016.

20. Herićko M., Živković A., The size and effort estimates in iterative development, Information and Software Technology, 50, 772-781, 2008.

21. Tabak B.A., Demirörs O., Efor Kestirim Doğruluğu için Tasarım Büyüklüğü ve Problem Büyüklüğü Karşılaştırılması, Ulusal Yazılım Mühendsiliği Sempozyumu (UYMS), İzmir-Türkiye, 25-28 Eylül, 2013.

22. Ren A., Yun C., Research of Software Size Estimation Method, International Conference on Cloud and Service Computing, Beijing-China, 154-155, November 4-6, 2013.

23. Mohagheghi P., Anda B., Conradi R., Effort Estimation of Use Cases for Incremental Large-Scale Software Development, 27th International Conference on Software Engineering, Saint Louis-USA, 303-311, May 15-21, 2005.

24. Ochodek M., Nawrocki K., Kwarciak K., Simplifying effort estimation based on use case points, Information and Software Technology, 53, 200-213, 2011.

25. Scientific Toolworks Inc. Understand User Guide and Reference Manual. http://scitools.com/documents/manuals/pdf/understand.pdf, Yayın tarihi Ekim 2015. Erişim tarihi Haziran 10, 2017.

26. Loper E., Bird S., NLTK: The natural language toolkit, In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, Philadelphia-USA, 1, 63-70, July 7, 2002.

27. Ayyıldız T.E., Koçyiğit A., Correlations between problem domain and solution domain size measures for open source software, 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2014), Verona-Italy, 81-84, August 27-29, 2014.

28. Ayyıldız T.E., Koçyiğit A., An Early Software Effort Estimation Method Based on Use Cases and Conceptual Classes, Journal of Software, 9 (8), 2169-2173, 2014.

29. Conte S.D., Dunsmore H.E., Shen V.Y., Software Effort Estimation and Productivity, Advances in Computers, 24, 1-60, 1985.

30. DeSanto C., Totoro M., Moscartelli R., Introduction to statistics, 9th Edition, Pearson, UK, 2010.

31. Çelik C., Bilge H.Ş., Feature Selection with Weighted Conditional Mutual Information, Journal of the Faculty of Engineering and Architecture of Gazi University, 30 (4), 585-596, 2015.

32. Akben S.B., Alkan A., Density-based Feature Extraction to Improve the Classification PErformance in the Datasets having Low Correlation between Attributes, Journal of the Faculty of Engineering and Architecture of Gazi University, 30 (4), 597-603, 2015.

33. Ryan, T.A., Joiner, B.L., Normal Probability Plots and Tests for Normality, Technical Report, Statistics Department, The Pennsylvania State University, 1976.

34. Miles J., Residual Plot, Wiley StatsRef: Statistics Reference Online, John Wiley & Sons, New York, A.B.D., 2014.

35. Hastings T.E., Sajeev A.S.M., A vector based approach to software size measurement and effort estimation, IEEE Transactions on Software Engineering, 24 (4), 337-350, 2001.

36. Conte S.D., Dunsmore H.E., Shen V.Y., Software Engineering Metrics and Models, Benjamin-Cummings Publishing Co., Redwood City, USA, 1986.

37. Foss T., Stensrud E., Kitchenham B., Myrtveit I., A simulation study of the model evaluation criterion MMRE, IEEE Transactions on Software Engineering, 29 (11), 985-995, 2003.

38. Shepperd M., MacDonell S., Evaluating prediction systems in software project estimation, Information and Software Technology, 54 (8), 820-827, 2012.

39. Jørgensen M., Experience with the Accuracy of Software Maintenance Task Effort Prediction Models, IEEE Transactions on Software Engineering, 21 (8), 674-681, 1995.

40. Tate G., Verner J., Software costing in practice, The Economics of Information and Software, R. Veryard, Butterworth-Heinemann, 101-126, 1990.

41. Kitchenham B.A., Pickard L.M., MacDonell S.G., Shepperd M.J., What accuracy statistics really measure, IEEE Proceedings Software, 148 (3), 81-85, 2001.

42. Rousseeuw P.J., Zomeren B.C., Unmasking outliers and leverage points, Journal of the American Statistical Association, 85, 633-639, 1990.

43. Acarlar I., Gamgam H., Çoklu Doğrusal Regresyonda Etkili Gözlem Gruplarının Saptanması İçin Kullanılan Tanı Yöntemlerinin Karşılaştırılması, TÜİK İstatistik Araştırma Dergisi, 7 (1), 83 -99, 2010.

44. Cook R.D., Detection of Influential Observations in Linear Regression, Technometrics (American Statistical Association), 19 (1), 15–18, 1977.

45. Ribu K., Estimating object-oriented software projects with use cases, Master's thesis, University of Oslo, Department of Informatics, Norway, 2001.

46. Zhou Y., Yang Y., Xu B., Leung H., Zhou X., Source code size estimation approaches for object oriented systems from UML class diagrams: A comparative study, Information and Software Technology, 56, 220-237, 2014.