Yazılım Kalite Metriklerinin Kıyaslanması: Örnek Bir Olay İncelemesi

Comparison of Software Quality Metrics: A Case Study

Alper Kıral[1][0000-0002-4018-0419] Tülin Erçelebi Ayyıldız[2][0000-0002-7372-0223]

[1,2] Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, 06790, Türkiye
[1]alperkiral@gmail.com [2]ercelebi@baskent.edu.tr

Özet. Yazılım projelerinde zaman kısıtlılığından dolayı gözden kaçırılan birçok nokta sonradan büyük sorunlar çıkarmaktadır. Bakım ve onarım sırasında yaşanabilecek sıkıntıları ortadan kaldırabilmek için yazılım projelerinin kalitesini ölçmek önem kazanmıştır.

Bu nedenle bu çalışmada yazılım projelerinin kalitesini ölçerek en çok dikkat edilmesi gereken ölçütlerin sıralanması örnek bir olay incelemesi ile sağlanmıştır. Örnek çalışmayı gerçekleştirebilmek için erişim kolaylığı nedeniyle 16 adet nesne yönelimli açık kaynak kodlu yazılım seçilmiş ve bu yazılımların kod karmaşıklık analizi yapılmıştır.

Yazılım projelerinin kalitelerini ölçmek için Chidamber and Kemerer"in (CK) yazılım kalite ölçütleri kullanılmış olup bu ölçütlerin değerleri "Understand" kod analiz aracı ile belirlenmiştir.

Elde edilen değerlerin literatürde belirtilen eşik değerlerini geçip geçmediği tespit edilerek 16 açık kaynak kodlu yazılım projesinde eşik değerini geçen CK yazılım kalite ölçütlerinin frekansları incelenmiş, SPSS istatistiksel analiz aracı ile ölçütler arasında önem dereceleri tespit edilmiş ve sonuçlar verilmiştir.

Sonuçlar değerlendirildiğinde yazılan kaynak kodda en dikkat edilmesi gereken ölçütlerin sıralaması LCOM (Lack of Cohesion in Methods), CBO (Coupling between Object Classes), DIT (Depth of Inheritence Tree), WMC (Weighted Methods per Class), RFC (Response for a Class) ve NOC (Number of Children) şeklinde olmuştur.

Anahtar Kelimeler: Yazılım Kalite Ölçütleri, CK ölçütleri, Karmaşıklık Analizi, Understand, SPSS.

Abstract. Lack of time during planning stages leads to overlooking some small but crucial details in software design projects. These points cause major problems later on along the life of the project. In order to avoid facing with many problems during maintenance phase, it is becoming much more important to measure the quality of the software projects.

For this reason, in this study, the quality of the software projects is measured and a ranking of the most important criteria is provided by a case study. Because of the ease of access to perform the sample work, 16 object-oriented open

source software was selected and the code complexity analysis of these software was performed.

The software quality criteria of Chidamber and Kemerer (CK) were used to measure the qualities of the software projects and the values of these criteria were determined by the "Understand" code analysis tool.

It was determined that the obtained values exceeded the threshold values stated in the literature. These frequencies of CK software quality criteria exceeding the threshold value in 16 open source software projects were examined and the importance ranks were determined with SPSS statistical analysis tool and the results were given.

When the results are evaluated, the most important criteria to be considered in the source code are Lack of Cohesion in Methods (LCOM), Coupling between Object Classes (CBO), Depth of Inherence Tree (DIT), Weighted Methods per Class (WMC) Class) and NOC (Number of Children), respectively.

Keywords: Project Quality Metrics, CK metric, Complexity Analysis, Understand, SPSS.

1 Giriş

Gelişen teknoloji ile birlikte yüksek işlem gücüne sahip bilgisayar donanımları düşük maliyetlerle üretilebilmektedir. Bu durum beraberinde daha kapsamlı yazılım ihtiyaları, daha büyük yazılım projeleri ve bu projeleri yürüten kalabalık ekipler getirmektedir.

Bir yazılım projesinin büyümesi demek; bakım-onarım masrafları, proje maliyeti, yazılım geliştirme zamanı vb. gibi değişkenlerin de büyümesi demektir. Eğer proje yönetim süreci doğru yürütülmezse, günümüzde olduğu gibi pek çok yazılım projesinin başarısızlıkla sonuçlandığı görülecektir. Kullanılamayan, müşteriler tarafından reddedilen, proje iptaliyle sonuçlanan, yüksek bakım-onarım maliyeti gerektiren yazılımların ülke ekonomisine verdiği zararlar da göz ardı edilemeyecek kadar büyük olabilmektedir. Örnek olarak, 2002 yılı verilerine göre başarısız olan yazılımların Amerikan ekonomisine yıllık zararı yaklaşık 59 Milyar $ olmuştur [1]. Ayrıca yazılım test araçları geliştiren Tricentis firmasının 2017 yılı verilerine göre de başarısız yazılımların dünya geneli yarattığı finansal zarar toplam 1.7 Trilyon $ civarındadır [13].

Yazılım kalitesinin ölçülmesi ile; yazılımın müşteri ihtiyaçlarına ne kadar hizmet edebildiği, geliştirici açısından yazılımın kolay anlaşılırlığı ve müdahale edilebilirliği, yazılımın yapısal kalitesi, arzu edilen kalitenin elde edilebilmesi için maliyet-fiyat dengesinin orantısı gibi anahtar faktörler projenin erken safhalarında değerlendirilebilir ve gerekli önlemler alınabilir.

Büyük yazılım projelerinde kod satırının, metot, sınıf, obje ve değişkenin tek tek takibinin ve kontrolünün kısa sürede yapılması güçtür. Bu ölçümleri çok kısa sürelerde gerçekleştirebilen çok sayıda ve çeşitte kod analiz araçları bulunmaktadır.

Bu çalışmada ilgili alanda daha önce yapılan çalışmalar incelenmiş, Java programlama dilinde yazılmış olan "Uluslararası Uzay ve Havacılık" sektörüne ait açık kaynak kodlu 16 adet nesne-yönelimli yazılımın kalite ölçümleri "Understand" statik kod

analiz aracı yardımıyla yapılarak ilgili metrik değerleri incelenmiştir. Elde edilen değerler yorumlanmıştır.

Çalışmanın 2. bölümünde literatür araştırmasına yer verilmiş, bölüm 3'te örnek olay incelemesi için kullanılan projelerden bahsedilerek eşik değerleri hesaplanmış ve son olarak da bölüm 4 'te elde edilen sonuçlar değerlendirilmiştir.

2 İlgili Çalışmalar

Yazılım kalitesini kalite özelliği ve kalite ölçütü olmak üzere iki ana başlık altında inceleyecek olursak, literatürde yaygın olarak kullanılan ISO 9126 uluslararası kalite standardının kalite gereksinimleri ve bu gereksinimler için tanımlanan kalite özellikleri Tablo 1'de verilmiştir [2].

Tablo 1. ISO 9126 Kalite Standardı için Kalite Gereksinimleri ve Kalite Özellikleri

Kalite Gereksinimleri	Kalite Özellikleri
İşlevsellik	Uygunluk, Doğruluk, Güvenlik vb.
Güvenilirlik	Olgunluk, Hata toleransı vb.
Kullanılabilirlik	Anlaşılabilirlik, Öğrenilebilirlik vb.
Verimlilik	Kaynak Kullanımı vb.
Bakım	Test Edilebilirlik, Sağlamlık vb.
Taşınabilirlik	Uyumluluk, Değiştirilebilirlik vb.

Bu gereksinimlere göre ölçümler yapabilmek için, yazılım alanında çalışan insanlar ölçüt kümeleri ortaya sunmuşlar. Bu kümeler statik ve dinamik olarak iki gruba ayrılabilmektedir. Statik metrikler yazılımın yapısıyla, dinamik metrikler ise yazılımın çalışma anındaki verileriyle ilgilidir.

Statik metrik kümelerinden en sık kullanılanlar arasında; ilk olarak 1974 yılında yazılım karmaşıklığını ölçmek amacıyla Thomas McCabe tarafından kendi adını verdiği McCabe Complexity Metric Suite öne sürülmüştür [4]. 1994 yılında Chidamber & Kemerer'in yayınladığı metrik kümesi CK Metric Suite ismiyle öne sürülmüş ve 6 metrikten oluşmaktadır [3]. Nesne yönelimli yöntemin temellerine dayanan MOOD metrik kümesi de kapsülleme, çok şekillilik ve mesaj aktarımı mekanizmalarına dayanmakta ve 6 metrikten oluşmaktadır [5].

Bu çalışmada, nesne yönelimli yazılımların kalıtım derinliği, sınıflar arası cohesion ve coupling durumu, sınıflara ait metot sayıları ve bu metotların başka sınıflardan çağırılma durumlarını daha etkili ölçebilme kabiliyetinden dolayı CK Metrikleri seçilmiş olup Java programlama dilinde yazılmış 16 açık kaynak kodlu proje, 6 adet CK metriği ile Understand kod analiz aracı kullanılarak ölçülmüştür.

Ancak literatürde bu ölçümler yapıldıktan sonra farklı farklı eşik değerleri üzerinden değerlendirmeler yapıldığı görülmüştür. Yine literatür araştırmalarında çalışılan projeye göre özel eşik değerleri hesaplamanın daha uygun olduğu görüşü daha geçerlidir [6].

3 Örnek Olay İncelemesi

Çalışma için güvenilirlik seviyesinin yüksek olacağı düşünüldüğünden "uzay ve havacılık" alanındaki projeler seçilmiştir. NASA, ESA ve SpaceX açık kaynak kodlu, JAVA programlama dilinde yazılmış nesne-yönelimli ve orta ölçekli yazılımlardır.

Bu yazılımlar, kuruluşun kendi sitesinden herkes tarafından erişilebilen yazılım kütüphanesi üzerinden [7] ve GitHUB'tan indirilmiştir. İlgili yazılımlar Tablo 2'de gösterilmektedir.

Tablo 2. Açık Kaynak Kodlu JAVA projeleri

Proje	KLOC	Proje	KLOC
CCSDS_MO_GraphicalEditor	11K	DAVEtools-master	15K
CCSDS_MO_TESTBEDS	25K	DERT-master	42K
CCSDS_MO_TRANS	14K	FEI-master	63K
GUSTO	13K	JavaGenes	32K
nmf-core	30K	jpf-core-master	114K
NMF_MISSION_OPS-SAT	32K	symbc-master	199K
NMF_MISSION_SOFTWARE_SI	12K	mct-master	124K
CCDD	75K	SpaceLaunchNow-Android-master	25K

Bu çalışmada kullanılan yazılım projelerinin analizi "Understand" statik kod analizi aracıyla gerçekleştirilmiştir. Scientific Tools INC. tarafından geliştirilen yazılım, kaynak kodunu analiz ederek kod hakkında; detaylı metrik raporu, akış diyagramı grafikleri, detaylı bağımlılık analizi gibi önemli bilgileri çıktı olarak sunmaktadır [8].

3.1 CK Metrikleri için Eşik-Değer Üretimi

CK Metrikleri için çeşitli çalışmalarda çeşitli eşik-değerleri önerilmiştir. Tablo 3'te bu değerlerden sık kullanılanlar gösterilmektedir.

Tablo 3. CK Metrik Eşik Değerleri

İlişkili Çalışmalar	LCOM	DIT	CBO	NOC	RFC	WMC
[9]	3	6	9	3	36	30
[10]	1	6	8	6	35	15
[11]	Düşük	4	8	6	35	11
[12]	20	2	14	2	44	20
[6]	Düşük	4	94	5	10	108

Tablo 3'te görüldüğü gibi herkes tarafından kabul gören sabit bir eşik değer bulabilmek çok güçtür ve CK Metrikleri için proje bazlı eşik-değer üretilmesi tavsiye edilmiştir. Statik kod analiz aracından elde edilen metrik sonuçları için, her bir metriğin sınıflara göre ölçüm değerlerinin ortalaması ve standart sapması bulunup toplanarak ilgili proje için bir eşik değer türetmek bir yöntem olarak kabul edilebilmektedir [6].

Kullanılan formül Denklem (1) de verilmiştir, T eşik değer, μ ortalama değer, Ω standart sapma değerlerini göstermektedir.

$$T = \mu + \Omega \quad (1)$$

Bu çalışmada tek bir proje yerine 16 adet yazılım projesi bulunduğundan, 16 proje için bulunan eşik değerlere yukarıdaki μ + Ω işlemi tekrar yapılarak genel bir eşik-değer kümesi oluşturulmuştur. Elde edilen değerler Tablo 3 ile birleştirilerek Tablo 4'te tekrar verilmiştir (Understand aracında LCOM değerleri yüzdelik olarak verildiği için tabloda da yüzdelik eşik değer kullanılmıştır).

Tablo 4. 16 Proje için CK Metrik Eşik Değerleri

İlişkili Çalışmalar	LCOM	DIT	CBO	NOC	RFC	WMC
[9]	3	6	9	3	36	30
[10]	1	6	8	6	35	15
[11]	Düşük	4	8	6	35	11
[12]	20	2	14	2	44	20
[6]	Düşük	4	94	5	10	108
Bu Çalışma	%86,8	2,9	10,8	14,8	62,4	25,6

3.2 Üretilen Eşik-değerler ile CK Metriklerinin Kıyaslanması

Elde edilen eşik değerleri, eşik değerlerini aşan sınıf frekansları ve proje içerisindeki tüm sınıf sayısına göre yüzdelik değerler Tablo 5'te verilmiştir.

Tablo 5. 16 JAVA Projesi için Eşik Değer Sınırını Aşma Frekansları

İlişkili Çalışmalar	LCOM	DIT	CBO	NOC	RFC	WMC	#Class
CCSDS_MO_Graphi.	13 (%14)	45 (%48)	11 (%12)	2 (%2)	30 (%32)	9 (%9)	93
CCSDS_MO_TESTBED	30 (%13)	18 (%8)	9 (%4)	1 (%0)	8 (%3)	18 (%8)	218
CCSDS_MO_TRANS	21 (%13)	37 (%24)	4 (%2)	0 (%0)	8 (%5)	7 (%5)	155
GUSTO	15 (%7)	4 (%2)	7 (%3)	0 (%0)	0 (%0)	9 (%4)	217
Nmf	14 (%4)	8 (%2)	25 (%8)	1 (%0)	1 (%0)	6 (%2)	332
NMF_MISSION_OPS-sa	42 (%9)	1 (%0)	5 (%1)	0 (%0)	4 (%1)	12 (%2)	482
NMF_MISSION_SOFT_	9 (%6)	0 (%0)	2 (%1)	0 (%0)	1 (%1)	7 (%5)	158
CCDD	92 (%10)	39 (%5)	82 (%10)	5 (%1)	9 (%1)	23 (%3)	842
DAVEtools	8 (%7)	15 (%14)	9 (%8)	0 (%0)	17 (%15)	8 (%7)	110
DERT	49 (%7)	132 (%19)	48 (%7)	4 (%1)	6 (%1)	22 (%3)	705
FEI	43 (%7)	2 (%0)	21 (%3)	0 (%0)	9 (%1)	33 (%5)	631
JavaGenes	24 (%4)	154 (%21)	73 (%10)	1 (%0)	8 (%1)	19 (%3)	717
Jpf-core	118 (%7)	232 (%13)	139 (%8)	9 (%0)	503 (%28)	81 (%4)	1810
symbc	242 (%10)	441 (%17)	241 (%10)	11 (%0)	675 (%26)	127 (%5)	2630
mct	249 (%10)	249 (%10))	155 (%6)	7 (%0)	75 (%3)	43 (%2)	2442
SpaceLaunchNow-	38 (%8)	23 (%5)	9 (%2)	0 (%0)	2 (%0)	10 (%2)	506

Tablo 5'e göre özellikle "High-Cohesion & Low Coupling" kuralına dikkat edilmesi gerektiği sonucuna varılabilmektedir.

Bu çalışmada projelerin geneli üzerinden DIT eşik değeri 2.9 olarak belirlenmiştir. Ancak kalıtım derinliği için genelde kullanılan 4-6 aralığında bir değer alınsa idi büyük oranda tüm projelerde DIT metrik değerini ihlal eden sınıf olmayacak idi. İncelenen 16 projenin tamamında DIT değeri çok nadiren 5 değerlerine çıkmakla birlikte, genelde 5'in altında gözlemlenmiştir. Proje bazlı 2.9 DIT eşik değerimizden kaynaklı frekans aşımını bu noktada göz ardı edilebileceği düşünülmektedir.

RFC ve WMC metrikleri için de yine proje bazlı çok farklı büyüklüklerde eşik değerler alabildiği Tablo 4'te görülebilmektedir. 16 projeden elde ettiğimiz RFC ve WMC eşik değerlerinin genel olarak çok fazla aşılmadığı da Tablo 5'te görülebilmektedir. Bazı tekil projeler yüksek aşım frekansına sahiptirler ve bu durum projeye özgü olarak değerlendirilebilir.

Yukarıdaki istisnai durumlar LCOM ve CBO için geçerli değildir. Bu nedenle cohesion ve coupling metrikleri daha önemli kabul edilebilir. Tablo 5'teki oranlar ve istisnai durumlar incelendiğinde metriklerin önem sıraları; LCOM, CBO, DIT, WMC, RFC ve NOC olacak şekilde sıralanabilmektedir.

4 Sonuçlar

Bu çalışmada yazılım projelerinin kalitesini ölçerek en çok dikkat edilmesi gereken ölçütlerin sıralanması örnek bir olay incelemesi ile sağlanmıştır. Örnek çalışmayı gerçekleştirebilmek için erişim kolaylığı nedeniyle 16 adet nesne yönelimli açık kaynak kodlu yazılım seçilmiş ve bu yazılımların kod karmaşıklık analizi yapılmıştır.

Yazılım projelerinin kalitelerini ölçmek için Chidamber and Kemerer'in (CK) yazılım kalite ölçütleri kullanılmış olup bu ölçütlerin değerleri "Understand" kod analiz aracı ile belirlenmiştir.

Sonuçlar değerlendirildiğinde yazılan kaynak kodda en dikkat edilmesi gereken ölçütlerin sıralaması LCOM (Lack of Cohesion in Methods), CBO (Coupling between Object Classes), DIT (Depth of Inheritence Tree), WMC (Weighted Methods per Class), RFC (Response for a Class) ve NOC (Number of Children) şeklinde olmuştur.

Yazılım geliştiriciler ve proje yöneticileri kod yazım esnasında en çok hata yapılan kısımları bildikleri takdirde erken safhada önlem alınmasının maddi manevi proje başarısını etkileyeceği düşünüldüğünden bu çalışmanın araştırmacılara yol göstereceği düşünülmekte olup, ileride yapılacak çalışmalarda proje sayısı artırılıp daha kapsamlı istatistiki çalışmalar yapılarak elde edilen veriler değerlendirilecektir.

Referanslar

1. National Institute of Standards and Technology Homepage, https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf, p:ES-3, Last Accessed 2018/08/07
2. Jung H., Kim S., and Chung C., "Measuring Software Product Quality: A Survey of ISO/IEC 9126", September/October 2004, IEEE Software, p:88-92.
3. S. Chidamber and C. Kemerer, "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476- 493, June 1994.
4. T. J. McCabe. A complexity measure. In ICSE '76: Proceedings of the 2nd international conference on Software engineering. IEEE Computer Society Press, 1976.
5. F. Brito e Abreu, G. Pereira, and P. Soursa, "CouplingGuided Cluster Analysis Approach to Reengineer the Modularity of Object-Oriented Systems," Proc. Euromicro Conf. Software Maintenance and Reeng., pp. 13-22, 2000.
6. S. Singh, M. Kaur, "Deriving And Validating Software Metrics Threshold Values For Design Errors" International Journal of Engineering Technology and Scientific Research Volume 2 Issue 2 (April 2016)
7. NASA Open Source Software, https://code.nasa.gov/, Last Accessed 2018/05/18
8. SciTools.com Features Page, https://scitools.com/features/ Last Accessed 2018/06/15
9. Antony P., Predicting Reliability of Software Using Thresholds of CK Metrics. International Journal of Advanced Networking, 1778–1785, 2013.
10. Zhou Y., Leung H., Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Transactions on Software Engineering, 771–789, 2006.
11. Mago J., Kaur P., Analysis of quality of the design of the object oriented software using fuzzy logic. International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings Published in International Journal of Computer Applications® (IJCA), 21–25, 2012.
12. A.D. Bakar, A. Sultan, H. Zulzalil and J. Din, 2014. Predicting Maintainability of Object-oriented Software Using Metric Threshold. Information Technology Journal, 13: 1540-1547.
13. Software Testing Tools for Continuous Testing Webpage, https://www.tricentis.com/wp-content/uploads/2018/01/20180119_Software-Fails-Watch_Small_Web.pdf, Last A ccessed 2018/08/07