Kodu İyileştirmeye Nereden Başlamalı? Bir Yazılım Metrik Yaklaşımı: Yazılım Kalite Risk Oranı

Furkan Palıgu, Savaş Öztürk, Nurhan Yağcı

Yazılım Test ve Kalite Değerlendirme Merkezi, TÜBİTAK, Kocaeli

{ furkan.paligu, savas.ozturk, nurhan.yagci}@tubitak.gov.tr

Özet. Kod gözden geçirme, yeniden yapılandırma, testlere başlama gibi kritik karar alma zamanlarında kodu iyileştirmeye nereden başlanacağı ya da en çok hangi modül ya da kod parçacıklarına dikkat edilmesi gerektiği önemli bir araştırma konusudur. Bu çalışmada, bir yazılım projesinde sadece kaynak kodu kullanarak yazılım metrikleri açısından en problemli metot ve sınıfların tespitine yönelik bir oran hesaplaması geliştirilmiştir. Karmaşıklık, Satır sayısı ve nesne yönelimli metrik gruplarından yaklaşık 20 adet metriğin ölçüm değerleri, eşik aşım sayısı ve eşiği aşma miktarları dikkate alınarak hesaplanan katsayıları kullanarak tümleştirilmiş ve en riskli yazılım bölümleri tespit edilmiştir. Bu çalışma sayesinde ekip liderleri sorumlu oldukları yazılım projelerinde kısa sürede etkin işbölümü ve organizasyon yaparak, geliştiricileri çok sayıda metriği ölçme ve değerlendirme yükünden kurtarmış olacaktır.

1 Giriş

Bir projenin yazılım kalite metrikleri ile statik analizleri yapıldığında genellikle çok fazla ve karışık sonuçlar ortaya çıkar. Sadece McCabe 150´den fazla metrik içermektedir [1]. Yazılım kalite ölçüm araçlardan alınan sonuçların yazılım geliştirici tarafından yorumlanması ve iyileştirme çalışmalarının başlaması zor ve zaman alıcıdır. Yazılımcı her zaman metriklerin tanımları ile aşina değildir ve projedeki birçok öğe arasından probleme etkisi yüksek olanları tespit etmek fazlaca gayret ister. Bu tespit sürecini hızlandırmak bütün metriklerden alınan sonuçların ortak bir paydada toplanması ile mümkündür. YKRO (Yazılım Kalite Risk Oranı) metriği, bütün metrik ölçüm sonuçları yorumlanarak oluşturulan birleştirici bir metriktir ve projede probleme etkisi en fazla olan öğelerin tespitinde kullanılması amaçlanmaktadır.

Bir projede her zaman gözden geçirme ve kalite arttırma faaliyetlerini detaylı olarak yürütecek zaman mevcut değildir. Proje yöneticisi, yazılımdaki kaliteyi sağlamak amacı ile ekipteki bazı bireyleri ekip arkadaşlarının geliştirdiği kodları gözden geçirmek üzere görevlendirir. Fakat bu çalışma genellikle hızlı ve verimsiz olarak tamamlanır. Yeterli zaman bulunsa dahi yazılımcılar çalışan kodlar üzerinde detaylı incelemeler yapmaya meyilli değildir. Dolayısı ile birçok önemli nokta gözden kaçmakta ve

çalışmalar bir formaliteye dönme eğilimi göstermektedir. Kalite metriklerine göre problemin yoğunlaştığı öğeleri deşifre etmek gözden geçirme faaliyetlerinde verimliliği arttıracaktır çünkü iyileştirme çalışmalarında gözden geçirilmesi gereken öncelikli öğeler aynı zamanda probleme etkisi en fazla olan öğelerdir, aynı şekilde ilk test edilmesi gereken öğeler problem çıkartma potansiyeli en yüksek olanlardır. Problemin odak noktalarının tespiti aynı zamanda öğelere işlem görme önceliklendirmesi sağlamış olur. Bu önceliklendirme sınırlı zaman içerisinde en verimli test ve gözden geçirmelerin yapılmasını böylece kısa bir süre içerisinde verimli bir iyileştirme gerçekleştirilmesini sağlar.

YKRO, ölçüm raporlarının hızlı ve verimli değerlendirilmesini sağladığı gibi proje ekibinin başarısı, problem oluşturan öğeler ve kalite metrik sonuçlarındaki problemin proje genelindeki dağılımı gibi proje yönetimini yakından ilgilendiren konular hakkında da bilgiler verir. Böylece proje yöneticisinin teknik detaya girmeden projenin gidişatı hakkında bilgi edinmesini sağlar. Problemin hangi öğelerde yoğunlaştığı bilgisinin proje yöneticisi tarafından sürekli olarak takip edilebilmesi bu öğelerden sorumlu olan yazılımcıların kalite iyileştirme faaliyetlerine daha fazla ilgi göstererek projenin sürdürülebilirlik ve bakım yapılabilirliğini arttırmasıyla sonuçlanır.

Bu kullanım tercih ve duruma göre daha verimli bir şekle getirilebilir, örneğin bir proje ekibi bütün YKRO değerlerini 10´un altına indirmek gibi bir hedef koyabilir. Böylece hiçbir metot ya da sınıf projedeki kalite ölçüm sorunlarının %10´undan fazlasını bünyesinde bulundurmuş olmaz. Proje düzenli bir yazılım kalite dağılımına sahip olur.

Bu makalede YKRO metriğinin prensipleri anlatılmış, benzer çalışmalardan bahsedilmiş ve çeşitli kodlar üzerinde yapılan metrik ölçüm sonuçlarından alınan değerler üzerinde örnekler gösterilmiştir.

2 İlgili Çalışmalar

Daha önce belirli metrikleri birleştiren çalışmalar yapılmıştır. Bu çalışmaların en çok bilineni Oman ve Hagemeister tarafından önerilen Maintainability Index (MI) çalışmasıdır. Maintainability Index bazı geleneksel metriklerin birleşimi ile oluşan ve bakım yapılabilirliği ölçmeyi amaçlayan bir metriktir. Halstead Metrik Grubu´dan efor ve hacim, McCabe´in çevrimsel karmaşıklığı ve kod satır sayısı metriklerinden oluşur. Bazı durumlarda bu birleşime yorum satır sayısı da dâhil olur. [2-3]

Kullanılan metriklerin seçimindeki gerekçe altta listelenen özellikleri sayısal olarak ifade edebilecek olan ölçüm metotlarını kendi bünyesinde birleştirmektir.

- Ne kadar değişken bulunduğu ve bu değişkenlerin nasıl kullanıldığı – efor ve hacim
- Programın karmaşıklığı – çevrimsel karmaşıklık
- Programın büyüklüğü – satır sayısı
- Programın bir insan tarafından anlaşılabilirliği – yorum satır sayısı

Maintainability Index için çeşitli farklılıklar içeren varyantlar da türemiştir [4-5-6] fakat bu çalışmaların hepsi genelinde aynı prensipleri takip eder.

Birleştirici bir metrik olma özelliğini taşıyan benzer bir yapı da sonar aracında kullanılan ve Ward Cunningham tarafından icat edilen Technical Debt kavramıdır[7]. 5 ölçütten elde edilen sonuçlara göre hesaplanır; Kod kopyalaması, yazılım kural ihlalleri, yorumlar, test kapsamı ve karmaşıklık. Zaman içerisinde hızlı ve verimsiz bir şekilde geliştirilen yazılımların bahsedilen ölçütlere göre oluşturduğu problemlerin çözülmesi için ne boyutta bir çalışma gücüne ihtiyaç olduğunun tespiti üzerine kurulmuştur.

Bu çalışmalar birçok ölçütten alınan değerlerin birleştirilip yorumlanması bakımından YKRO ile aynı çizgidedir. Fakat bu çalışmaların amacı ölçülebilir bir rakam ile problemin boyutunu sergilemektir. YKRO´un amacı ise problemin ne boyutta olduğunu göz önüne almaksızın var olan problemin hangi öğeler üzerinde yoğunlaştığını gösterip problemin hızlı çözülmesi için bir yol gösterici olmaktır.

YKRO problemin boyutunu işaret eden yapılar ile kullanıldığında daha verimli olur. Bir alternatif değil bir tamamlayıcıdır. Bir kullanıcı, kalite metrik sonuçlarından problemin varlığı ve ciddiyeti hakkındaki bilgileri aldığında bir sonraki aşamaya yani bu problemin çözümü üzerine stratejiler geliştirilmesine geçer. Burada YKRO´un sorunlu öğeleri hedef göstermesi strateji çalışmalarındaki verimliliği arttırır. İdeal bir kalite değerlendirme raporunda problemin boyutu, yeri ve çözümü üzerine öneriler olmalıdır. Böylece yazılım kalite değerlendirme raporu, yazılımcının hatalarını açığa vuran bir kavram olmaktan kurtulup proje ekibinin büyük bir yardımcısına dönüşür. YKRO metriğinin en temel amacı da budur.

3 Önerilen Çalışma

YKRO değeri bir metodun proje kalite ölçüm problemlerine yaptığı katkının yüzdelik ifadesine tekabül eder. Örneğin bir sınıfın YKRO değeri 20 olarak hesaplanmış ise, o sınıftaki kalite problemlerinin çözümü proje genelindeki sınıf problemlerin %20 azalması demektir. Buradaki amaç problemin büyüklüğünü değil var olan problemin metot ve sınıflar üzerindeki dağılımını göstermektir.

Bir öğenin projedeki metrik kalite ölçümlerinde ortaya çıkan problemlerin ne kadarını oluşturduğunu tespit etmek, o metodun her metrik ölçümünden aldığı sonuçların ve projenin geri kalan metotlarının aynı ölçümlerde verdiği sonuçların değerlendirilmesi ile olur. McCabe IQ aracında ölçüm birçok metrik ile yapılır ve her metriğin eşik değeri birbirinden farklıdır. Bu sebeple farklı metrik ölçüm sonuçlarının doğrudan bütünleştirilmesi mümkün değildir. Her metriğin ayrı olarak ele alınması ve verdiği sonucun belirli katsayılar ile işlenerek genelde birleştirilmesi gerekir.

3.1 YKRO Hesaplama

Bir metot ya da sınıfın YKRO değeri Formül (1) ve Formül (2) ye göre hesaplanır.

$$Td_j = \sum (MV_{ij} - Thr_j) \quad (1)$$

$$YKRO_i = \sum ((MV_{ij} - Thr_j) / Td_j * 100) * C_j \quad (2)$$

Formül (1) ve Formül (2) de i metot ya da sınıf numaralandırıcısıdır, j metrik tipi, Tdj metot ya da sınıfın risk değeri, MVij metrik ölçüm sonucu, Thrj seçilen metriğin eşik değeri ve Cj seçilen metriğe verilen katsayıdır. MVij - Thrj değerinin Thrj´in MVij den büyük olması durumunda 0 olarak kabul edilmesi gereklidir.

Seçilen metrikler, tercih edilen eşik değerleri ve belirlenen metrik önem katsayıları Tablo 1´de gösterilmiştir. Metrikler seviyelerine göre iki gruba ayrılmıştır; metot seviyesindeki metrikler ve sınıf seviyesindeki metrikler. İki grupta da katsayıların toplamı 1´i verir.

Metrik önem katsayısı atamasında 3 farklı yöntem kullanılır; bu yöntemler metrik önem katsayısı başlığı altında detaylı olarak anlatılmıştır. Katsayılar proje tipine ya da tercihlere göre atanabilir. Bu çalışmada kullanılan metrik önem katsayıları yazılımın türü ve önceki ölçüm değerlerine göre tercihler doğrultusunda atanmıştır.

Tablo 1. Seçilen Yazılım Metrikleri [8]

Metrik	Metrik Kodu	Metrik Seviyesi	Eşik	Metrik Önem Katsayısı
Çevrimsel Karmaşıklık	v(G)	Metot	10	0.20
Esas Karmaşıklık	ev(G)	Metot	4	0.20
Modül Tasarım Karmaşıklığı	iv(G)	Metot	7	0.10
Genel Veri Karmaşıklığı	gdv(G)	Metot	4	0.05
Kod Satır Sayısı	Code	Metot	30	0.20
Yorum Satır Sayısı	Comment	Metot	10	0.10
Boş Satır Sayısı	Blank	Metot	10	0.10
Karışık Satır Sayısı	mixed	Metot	10	0.05
Maksimum Çevrimsel Karmaşıklık	MAXV	Sınıf	10	0.20
Maksimum Esas Karmaşıklık	MAXEV	Sınıf	4	0.20
Ortalama Çevrimsel Karmaşıklık	AVGV	Sınıf	10	0.10
Toplam Çevrimsel Karmaşıklık	SUMV	Sınıf	70	0.05
Kalıtım Ağacı Derinliği	DIT	Sınıf	7	0.10
Bütünlük Kaybı	LOCM	Sınıf	75	0.05
Bir Sınıf İçin Çağrı Sayısı	RFC	Sınıf	100	0.05
Sınıf İçindeki Metot Sayısı	WMC	Sınıf	14	0.05
Bağımlı Sınıf Sayısı	CBO	Sınıf	2	0.05
Herkese Açık Veri	pubdata	Sınıf	0	0.05

Tablo 2. Çevrimsel Karmaşıklık için Örnek YKRO Hesaplaması

v(G) Eşik Değeri	10			
v(G) Metrik Önem Katsayısı	0.125			
	Metot A	**Metot B**	**Metot C**	**TOTAL**
Ölçülen v(G)	15	23	7	
Fark = Ölçülen – Eşik Değeri	5	13	0	18
*Ayarlı Fark = (Fark / Toplam Fark) *100*	27.8	72.2	0	%100
*Ağırlıklı Değer = Ayarlı Fark * Metrik Önem*	**3.48**	**9.02**	**0**	**%12.5**

Tablo 2´de YKRO´un örnek bir hesaplanması detaylı olarak gösterilmiştir. Örnekte 3 farklı metot için çevrimsel karmaşıklık metriğinin YKRO etkisi hesaplanmaktadır. Tablodaki hesaplamada B metodu %10 değeriyle çevrimsel karmaşıklık bakımından eşik aşımına etkisi en yüksek olan metot olarak tespit edilmiştir. Bu hesaplama bütün metot seviyesindeki metrikler için yapıldığında metotların her metrik için alacakları değerlerin toplamı o metotların YKRO değerleri olacaktır.

YKRO değeri her metot ve sınıf için hesaplandıktan ve değerler azalan düzende sıralandıktan sonra sınıf ve metotlar olmak üzere iki riskli grup oluşturulur. Her seviyedeki YKRO toplamları her zaman 100 çıkacaktır.

3 farklı metot ve bu metotların bazı metriklerin ölçümünden alınan sonuçları Tablo 3´de listelenmiştir. Metot D ve Metot E için sadece Kod Satır Sayısı metriği eşik değerinin üzerinde kalmıştır. Dolayısı ile bu metotlar için YKRO değerleri sadece Kod Satır Sayısı metriğinden alacakları değerler ile belirlenecektir. Metot F için bütün ölçüm değerlerinin metrik eşik değerleri ile aynı olduğu gözlenmektedir. Bu durumda eşik aşılmadığı için YKRO değeri 0 olarak hesaplanır. Görüldüğü üzere Metot F´den alınan ölçüm sonuçları Metot D´den alınan sonuçlara kıyasla daha yüksektir, fakat metriklerden birinin eşik değerinin aşılması Metot D´in Metot F´den daha kötü bir sonuç almasıyla sonuçlanmıştır. Görülmektedir ki kullanılacak olan eşik değerlerinin belirlenmesi proje için önemli bir karar aşamasıdır.

Tablo 3. Karşılaştırmalı YKRO Örneği

	LOC	ev(G)	v(G)	iv(G)	YKRO
Esik Değerleri	30	4	10	7	7
Metot D	32	3	3	3	11.5
Metot E	45	3	3	3	25.4
Metot F	30	4	10	7	0

3.2 Örnek Çalışma

Yazılım Genel bilgileri Tablo 4´de verilen proje 14240 satır sayısı ile orta büyüklükte kabul edilen bir Java projesidir[9-10]. Yazılım kalite metrikleri ölçümleri McCabe IQ aracı kullanılarak alınmıştır. Projenin ölçümleri üzerinde YKRO değerleri önceki bölümde verilen Formül (2)´ye göre Tablo 1´deki metrik önem katsayıları kullanılarak hesaplanmıştır.

Tablo 4. Örnek Proje Genel Değerleri

Parametre	**Miktar**
Satır sayısı	14240
Sınıf sayısı	129
Metot sayısı	741

Hesaplanan metot YKRO değerlerinin ilk 7´si Tablo 5´de azalan sırada listelenmiştir. Sadece birkaç metot üzerinde iyileştirme çalışması yapılması projeyi düzgün ve dengeli bir yapıya sokacaktır. 741 tane metodu olan bu projede 3 metodun yazılım kalite ölçüm problemlerindeki etkisi %40 civarındadır (3 metodun YKRO değerlerinin toplamı %40 civarındadır). YKRO´ın projedeki risk kaynakları olarak işaret ettiği bu metotların YKRO değerlerinin yanında metrik ölçüm sonuçları de verilmiştir. Böylece kod iyileştirme çalışmalarına nereden ve nasıl başlanacağı basit bir tablo ile ifade edilmiştir.

Tablo 5. Metot YKRO Sıralanmış Listesi

		Boyut	Karmaşıklık		
Metot Adı	**YKRO**	**LOC**	**ev(G)**	**v(G)**	**iv(G)**
RuleParser.parseRuleTokens()	**24.46**	360	67	101	71
AuctioneerAgent.EvaluateMessages.onTick()	**11.03**	272	18	45	35
RobotAgent.ListenNReply.onTick()	**6.68**	154	10	37	30
TerrainAgent.PaintArena.SimpleEnough()	**3.66**	68	18	18	17
RobotAgent.CreateSchedule_RRT.action()	**3.38**	129	1	28	19
AuctioneerAgent.StartAnAuctionOnce.action()	**3.33**	103	13	20	11
TerrainAgent.PaintArena.BuildTransitions()	**3.05**	130	1	34	9

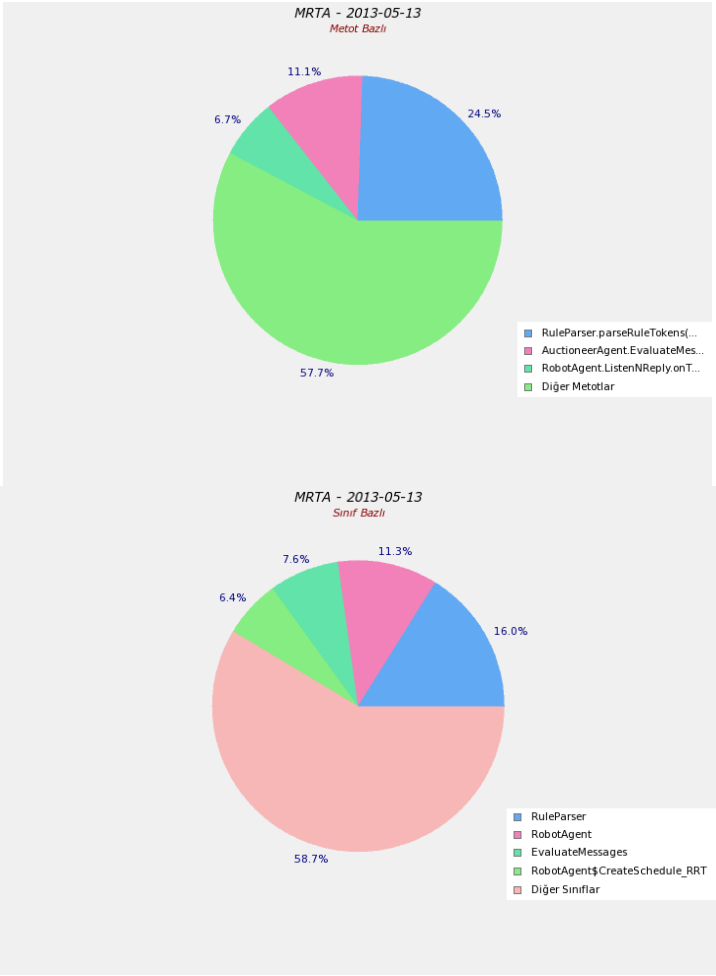
Hesaplanan sınıf YKRO değerlerinin ilk 7´si Tablo 6´de azalan sırada listelenmiştir. Ölçüm sonuçları kullanılan bu projedeki sınıfların metotlardan daha kararlı bir yapıya sahip olduğu gözlense de 127 sınıftan ilk 7´sinin ölçümlerin sınıfsal problemlerin %55,87´lik bir kısmını oluşturması genelde bir kararsızlığın söz konusu olduğunu gösterir. Kod iyileştirme çalışmalarına öncelikli olarak girmesi gereken sınıflar RuleParser ve RobotAgent sınıflarıdır.

Tablo 6. Sınıf YKRO Sıralanmış Listesi

Sınıf Adı	**YKRO**	**Sum v(G)**	**Avg v(G)**	**Max v(G)**	**Max ev(G)**	**DIT**	**RFC**	**WMC**	**CBO**	**LOCM**	**Pub Data**
RuleParser	**16.01**	112	18.6	101	67	2	6	6	0	66	57
RobotAgent	**11.32**	328	3.69	18	5	2	133	89	0	96	1091
EvaluateMessages	**7.56**	46	23	45	18	2	2	2	0	0	0
RobotAgent$Create	**6.41**	28	28	28	1	2	1	1	0	0	0
AuctioneerAgent	**4.91**	20	20	20	13	2	1	1	0	0	0
TerrainAgent$Paint	**4.83**	178	4.94	34	18	2	36	36	0	100	0
ListenNReply	**4.77**	23	19	37	10	2	2	2	0	10	0

Öğelerin metot ve sınıf YKRO değerlerleri toplamlarının 100 vermesi, sonuçların pasta grafikleri ile gösterimini de uygun kılar. Şekil 1 projenin metot ve sınıf YKRO

pasta grafiklerini gösterir. İki grafikte de değeri %5´in üzerinde kalan öğeler grafiğe dâhil edilmiş diğer öğelerin değerleri ise toplanarak tek bir dilim şeklinde gösterilmiştir. Bu grafiklerin daha basit ve ihtiyaç duyulan bilgileri göstermek üzere tasarlanmasından ileri gelmektedir. Proje ekibi başka bir değer belirlemek ya da sıralamadan istediği sayıda öğe göstermek gibi farklı tercihlerde bulunabilir.



Şekil 1. Örnek Proje YKRO Pasta Grafikleri

3.3 Metrik Önem Katsayısı

YKRO Katsayıları 3 farklı yöntem ile atanabilir. 'Tercih Yöntemi' YKRO metriğinin ilk hesaplamalarında kullanılmış, 'Otomatik Atama Yöntemi' ve 'Karma Yöntem' ise zaman içerisinde farklı tür projelerden farklı tür sonuçlar alınmasına binaen çıkan ihtiyaçlar ışığında geliştirilmiştir.

- **Tercih Yöntemi**: proje ekibi tecrübeleri ile proje tipini ve şartlarını değerlendirip hangi metriğin daha büyük bir öneme sahip olduğuna karar verir ve katsayı atamaları alınan bu kararlara göre yapılır.
- **Otomatik Atama Yöntemi**: Bu yöntemde metriğin eşik değerini en yüksek derecede geçen metodunun değeri göz önüne alınır. Temelde, bir metrik ile ilgili olan problemlerin daha büyük olması o metriğin proje genelindeki problemlere yapacağı etkinin daha büyük olmasına yol açacağı düşüncesine dayanır.
- **Karma Yöntem**: Bu yöntem tercih ve eşik otomatik atama yöntemlerinin bir bileşkesidir. Eğer metriklerin önem katsayıları metotların eşik aşım miktarına göre atanırsa, proje için önemi çok yüksek olmayan bir metrik kritik bir konumda kabul edilebilir ve bu proje ekibini yanlış yönlendirebilir. Öteki taraftan, iki metriğin de aynı ya da yakın öneme sahip olduğu bir durumda metriklerden birinin eşiği çok fazla diğerinin ise kritik kabul edilemeyecek derecede aşması metotlar arasında dengesiz bir dağılıma yol açar.
- Karma hesaplama yönteminde, katsayıların belirli bir kısmı atanacağı metriğin eşik aşımına geri kalan kısmı ise proje ekibinin belirleyeceği önem durumuna göre atanır.

Katsayı atamalarının farklı metotlar kullanılarak yapılmasının ölçüm sonuçlarına ne tür değişiklikler yapabileceğini daha iyi görmek için, geliştirme aşamasında olan bir projenin kodları McCabe IQ aracında ölçüldü. Tablo 1 de verilen metriklerden alınan sonuçlara göre metrik önem katsayıları, verilen 3 farklı metot ile atanmak suretiyle 3 farklı YKRO hesabı yapıldı. Bu projenin yapılan deneyde tercih edilmesinin sebebi metotlarından birinin kod satır sayısı ölçüm değeri 814 iken karmaşıklık grubu metriklerinin ölçümlerinden alınan sonuçların eşiğin çok üzerinde olmamasıdır.

Projenin kalite metrik ölçüm sonuçlarından alınan bilgiler doğrultusunda, her metrik için metotlardan alınan maksimum değer ve metrikler için tercih edilen önem katsayıları Tablo 7´de gösterilmiştir.

Tablo 7´de görüldüğü gibi karmaşıklık metriklerinde eşik değerinin en üstünde alınan değer iv(G) metriğine aittir. Eşik değeri 7 olan iv(G) metriği için %342´lik bir aşım söz konusudur. Aynı projede satır sayısı metrikleri tarafından yapılan en büyük eşik aşımı %2713 ile kod satır sayısında gözlenmektedir. Kod satır sayısındaki bu astronomik fazlalık tercihsel önem katsayısı atama yönteminde tespit edilemez çünkü her metrik genel yüzdede kendisine verilen katsayı kadar etki gücüne sahiptir. Otomatik katsayı atama yönteminde ise eşik aşım miktarı ile doğru orantılı olan bir hesaplama olacağından dolayı bu satır sayısı değerine sahip metodun YKRO değeri de oldukça yüksek olacaktır.

Kod satır sayısı metriği bazı projelerde yüksek bazılarında ise düşük olma özelliği göstermektedir. Örneğin donanım ağırlıklı bir projede satır sayıları genellikle yüksek

olacaktır. Karakteristik özellikleri yüksek olan bir projede metrik önem katsayılarının elle atanması daha iyi sonuçlar verebilir. Fakat genel yazılım karakteristikleri gösteren projelerde eşik aşım miktarı da göz önünde bulundurulmalıdır. Buna göre proje ekipleri kendilerine uygun olan yöntemi seçebilir ya da kendi yöntemlerini geliştirebilirler.

Tablo 7. Örnek Proje Ölçüm Sonuçları Maksimum Değerleri

Metrik	Metrik Kodu	Eşik	Maksimum Değer	Tercih Edilen Katsayı
Cevrimsel Karmaşıklık	v(G)	10	31	0.20
Esas Karmaşıklık	ev(G)	4	19	0.20
Modül Tasarım Karmaşıklı-	iv(G)	7	24	0.10
Genel Veri Karmaşıklığı	gdv(G)	4	19	0.05
Kod Satır Sayısı	Code	30	814	0.20
Yorum Satır Sayısı	Comment	10	60	0.10
Bos Satır Sayısı	Blank	10	118	0.10
Karışık Satır Sayısı	Mixed	10	8	0.05

Tablo 8 de metotların her üç katsayı hesaplama yönteminden ortaya çıkan sonuçlar gösterilmiştir. Metotlar üç YKRO´dan en az birinde ilk beşe giren metotlardır.

Tablo 8. Karşılaştırmalı YKRO Katsayı Atama Sonuçları

	YKRO Tercih	YKRO Otomatik	YKRO Karma
Metot A	2.93	8.77	5.85
Metot B	3.28	2.93	3.11
Metot C	2.38	1.34	1.86
Metot D	2.58	0.7	1.65
Metot E	2.33	1.45	1.89
Metot F	0.7	2	1.35
Metot G	0.66	1.91	1.28

4 Sonuç ve Değerlendirmeler

Yazılım kalite değerlendirme metrikleri ile yapılan ölçümlerin değerlendirilmesi yazılım verimliliğinin arttırılmasında kullanılan önemli yöntemlerden biridir. Bu metriklerden alınan sonuçların daha hızlı değerlendirilmesini sağlamak amacı ile birleştirici bir metrik olan YKRO metriği tanımlanmıştır. Bu metrik yeniden yapılandırılması gerekli olan metot ve sınıfları açıkça ortaya koyar. YKRO metriğinin her metot üzerinde ve ölçümlerde kullanılan her metrik üzerinde nasıl hesaplandığı bir örnek üze-

rinden anlatılmıştır. Hesaplamalara, kullanılan metriklerin ne ölçülerde katkı sağlayacağını belirlemek için metrik önem katsayısı ortaya konmuş ve bu katsayının belirlenmesinde kullanılabilecek olan yöntemler örnekler ve bu örneklerden alınan sonuçlar ile beraber gösterilmiştir. YKRO metriği kullanılarak sadece kaynak kod analizi ile çok kısa bir süre içerisinde yazılımda ilk olarak iyileştirilmesi gereken yerler belirlenebilmektedir. Böylelikle gerek kod gözden geçirme sürecinde, gerekse testlere başlamadan önce sorun çıkması muhtemel kod parçaları öncelikli olarak incelemeye alınabilmektedir. Daha sonraki çalışmalar metriklerin önce kendi karakteristik özelliklerine göre sınıflanıp, bu sınıflardan alınan sonuçların hesaplamalarda kullanılması ile ilgili olacaktır.

Teşekkür - Yazarlar, bu çalışmanın gerçekleştirilmesi için kaynak ve destek sağlayan TÜBİTAK BİLGEM Yazılım Test ve Kalite Değerlendirme Merkezi'ne teşekkür eder.

Kaynaklar

1. McCabe Software http://www.mccabe.com
2. Oman, P.W.; Hagemeister, J.; and Ash, D., A Definition and Taxonomy for Software Maintainability, Technical Report #91-08-TR, Software Engineering Test Laboratory, University of Idaho, Moscow, ID, 1991.
3. Oman, P.W. and Hagemeister, J., (1992) Metrics for Assessing a Software System's Maintainability, Proceedings of the Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 337-344.
4. Coleman, D., Assessing Maintainability, Proceedings of the Software Engineering Productivity Conference 1992, Hewlett-Packard, Palo Alto, CA, 1992, pp. 525-532.
5. Coleman, D.; Ash, D.; Lowther, B.; and Oman, P.W., Using Metrics to Evaluate Software System Maintainability, IEEE Computer, 1994, 27(8), pp. 44-49.
6. Oman, P.W. and Hagemeister, J., Constructing and Testing of Polynomials Predicting Software Maintainability, Journal of Systems and Software, 1994, 24(3), pp. 251-266.
7. Sonar Source, "Evaluate Your Technical Debt" http://www.sonarsource.org/evaluate-your-technical-debt-with-sonar
8. McCabe Software, "All Metrics Thresholds in McCabe IQ" http://www.mccabe.com/pdf/McCabe%20IQ%20Metrics.pdf
9. Emin B., Fatih Y., Önder S. "Yazılım Projelerinde Büyüklük Tahmini", Akademik Bilişim 2013, Akdeniz Üniversitesi, Antalya, 2013
10. Total Metrics, "'Small Project', 'Medium-Size Project' and 'Large Project' What Do These Terms Mean?" http://www.totalmetrics.com/function-points-downloads/Function-Point-Scale-Project-Size.pdf