Açık Kaynak Yazılım Seçimi için İki Boyutlu Değerlendirme Metodu

Nebi Yılmaz, Kıvanç Dinçer

Bilgisayar Mühendisliği Bölümü
Hacettepe Üniversitesi, Ankara

{yilmaz@cs, kivanc.dincer@}hacettepe.edu.tr

Özet. Son yıllarda açık kaynak yazılımların popülerliğinin artması, birbirine alternatif olarak pazara sunulan bu tür yazılımların sayısının hızla artmasına sebep olmuştur. Ancak açık kaynak yazılımların kalitesinin potansiyel kullanıcılar tarafından nasıl değerlendirilebileceğine ışık tutan akademik çalışmalar sınırlı sayıdadır. Dolayısıyla alternatifler arasından ihtiyaç sahiplerinin kalite gereksinimlerini en iyi karşılayanın seçilmesi problemi araştırılması gereken cazip bir problem haline gelmiştir. Bu bildiride literatürde mevcut çalışmalar kendi katkılarımızla sentezlenerek oluşturulmuş ve kod-tabanlı ve toplum-tabanlı olmak üzere iki açıdan ürün kalitesini değerlendirme imkânı sunan kapsamlı bir metot tanıtılmaktadır. Kod–tabanlı değerlendirme yapılırken, yazılımın kaynak kodları analiz edilerek ISO 25010 yazılım kalite modeline göre belirlenen ve kullanıcı için öncelikli içsel öznitelikler uygun metriklerle ölçülmektedir. Toplum-tabanlı değerlendirme yapılırken ise elektronik posta listeleri, problem (hata) raporları ve sıkça sorulan sorular vb. gibi tarihsel verilerden elde edilen metrikler kullanılmaktadır.

Anahtar kelimeler: Açık Kaynak Yazılım, Yazılım Değerlendirme/Seçme, Yazılım Kalite Değerlendirme, ISO 25010 Modeli, Yazılım Kalite Modelleri.

Abstract. Increased popularity of open source software has led to a considerable proliferation of alternative software. However, this being the case, an evident lack of academic studies that would contribute to the evaluation of open source software for organizations has turned the process of selecting the most suitable open source software product that meets the users' quality requirements into an appealing research problem. In this study, a comprehensive method to solve this problem has been obtained from the synthesis of existing studies in the literature with our contribution, which enables product evaluation using both code-based and community-based assessment. In order to perform code-based evaluation, internal attributes of the latest quality model ISO 25010 were used, and the proper metrics were employed in an attempt to measure these attributes. Furthermore, to perform community-based evaluation, metrics obtained from historical data such as e-mailing lists, program reports, frequently asked questions, etc. were utilized.

Keywords: Open Source Software, Software Evaluation/Selection Methods, Software Quality Evaluation, ISO 25010 Model, Software Quality Models.

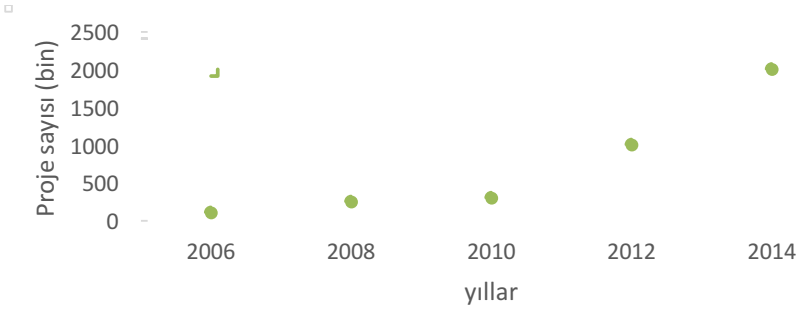
1 Giriş

Açık kaynak yazılımlar, kaynak kodları özel bir telif hakkı lisansı (copyright) ile herkesin incelemesine, kullanımına ve dağıtımına açılan böylece kullanıcıya yazılımı değiştirme özgürlüğü sunan ve dünyanın her tarafından bilişim uzmanlarınca imece yöntemi ile endüstri standartlarında geliştirilen yazılımlardır [1].

Açık kaynak yazılım kullanımı son yıllarda büyük oranda artmıştır. Bu yazılımların tercih edilmesinin sebebi geçmişten günümüze farklılık göstermektedir. Geçmiş yıllarda açık kaynak yazılım kullanımının ana sebebi mevcut kod tabanının ihtiyaca göre değiştirilip tekrar kullanımının kaynak ve zaman tasarrufu sağlaması iken (tekrar kullanılabilirlik), son yıllarda bu sebeplere ek olarak açık kaynak yazılımların yüksek kaliteli, güvenli ve güvenilir olarak algılanmaya başlanması da kullanımdaki bu artışı hızlandırmıştır. Böyle algılanmasındaki temel sebep ise bu yazılımların birçok geliştiricinin dikkatli incelemesinden geçmiş ve dolayısıyla hatalarından arındırılmış olduğunun kabul edilmesidir.

Avrupa Birliği, UNESCO, Dünya Bankası gibi kuruluşlar yukarıda zikredilen gerekçelerle açık kaynak yazılımların kullanımını önermektedirler. Nitekim Almanya, İspanya, Meksika, Brezilya, Çin, Kore, Hindistan gibi birçok ülke, kamu kurumlarında açık kaynak kodlu yazılımlarının kullanımını benimsemiş ve bilgi toplumu stratejilerinin bir parçası yapmışlardır [2].

Şekil 1'de verilen Black Duck Software ve North Bridge Venture organizasyonlarının raporuna göre, özellikle 2010 yılından sonra açık kaynak yazılımlara ilgi dikkat çekecek şekilde artmıştır. Geçmiş yıllarda Michael Skok: "Yazılımlar dünyayı ele geçiriyor (İngilizcesi: Software is eating the world)." demiştir. Fakat günümüzde yine aynı organizasyonların güncel raporuna göre bu algı değişmiştir ve "Açık kaynak yazılımlar yazılım dünyasını ele geçiriyor (İngilizcesi: Open source software is eating the software world)" haline dönüşmüştür [3].



Şekil 1. Açık kaynak yazılımı kullanımının yıllara göre değişimi

Açık kaynak yazılım ürünlerinin popülerliği bu derece artmışken, ticari yazılımlarda olduğu gibi en büyük problem bu yazılım ürünlerini değerlendirme ve seçme iş-

lemidir [4]. Akademik ve endüstri alanında ticari yazılımlar kadar olmasa da, açık kaynak yazılımların değerlendirilmesi ve seçilmesi için birkaç metot önerilmiş fakat hala standartlaştırılmış bir metot olmadığı için yapılan çalışmalar eksik ve yetersiz kalmıştır. Önerilen bazı genel metotlar [5]'deki gibi açık kaynak yazılımlar için kullanılabilir olmasına rağmen özellikle bu ihtiyaç için geliştirilmemiştir. Bu eksiklikten dolayı ihtiyaç sahipleri ihtiyaçlarını karşılayan açık kaynak yazılım ürünlerini subjektif değerlendirmelerle veya tavsiyeler üzerine seçmektedirler [6].

Bu çalışmada bu konudaki eksikliği giderebilmek için literatürdeki çalışmalardan bir sentez yapılarak kendi katkılarımızla beraber bu süreçte kod-tabanlı (code-based) ve toplum-tabanlı (community-based) olmak üzere iki açıdan değerlendirme imkânı sunan bir metot geliştirilmesi hedeflenmiştir. Kod–tabanlı değerlendirme yapılırken, yazılımın kaynak kodları analiz edilerek ISO 25010 kalite modelinde listelenen içsel öznitelikler arasından ihtiyaç sahibi tarafından önemli görülenler seçilerek uygun metriklerle ölçülmektedir. Toplum-tabanlı değerlendirme yapılırken ise açık kaynak yazılım ürünleriyle birlikte sağlanan kaynak kod ve geliştirme süreci hakkındaki tarihsel verilerden faydalanarak elektronik posta listeleri, problem (hata) raporları ve sıkça sorulan sorular vb. gibi tarihsel verilerden elde edilen metrikler değerlendirilmektedir.

Bölüm 2'de konuyla ilgili çalışmalardan bahsedilecek, Bölüm 3'te bu çalışma kapsamında geliştirilen bu kapsamlı metot tanımlanacak ve ilgili süreç adımları açıklanacaktır. Bölüm 4'te geliştirdiğimiz metodun iki açık kaynak yazılım ürününe uygulanmasını içeren doğrulayıcı bir vaka çalışması (case study) takdim edilecek ve elde edilen bulgular analiz edilecektir. Bölüm 5'de ise sonuçlar ve gelecek çalışmalarla bildiri sonlandırılacaktır.

2 İlgili Çalışmalar

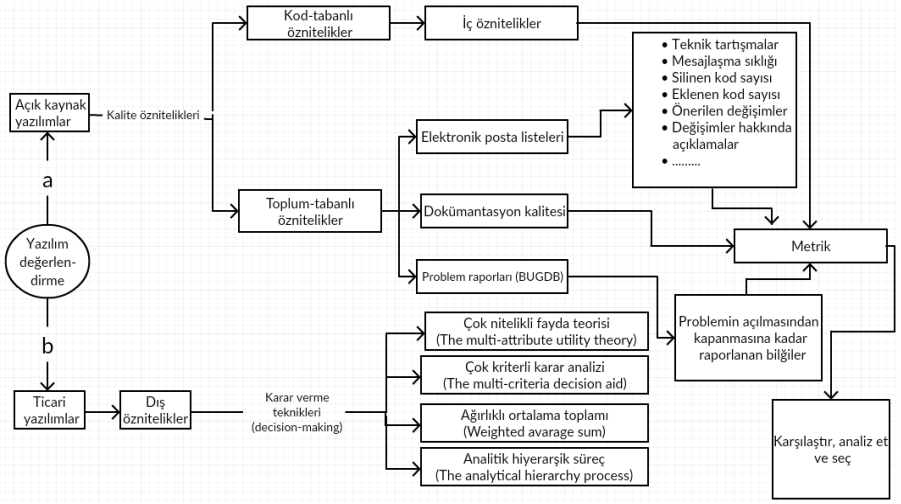
Literatürde açık kaynak yazılımların kalite özelliklerinin değerlendirilmesine ilişkin az sayıda çalışma bulunmuş, fakat bu çalışmalarda önerilen metotları derinlemesine karşılaştıran ve analiz eden bir çalışma bulunmamaktadır. Bu alanda en önemli çalışmalardan bir tanesini Wheeler yapmıştır [7]. Çalışmasında açık kaynak yazılımların değerlendirilme sürecini ticari yazılımların değerlendirilme süreci ile kıyaslamıştır ve açık kaynak yazılımların değerlendirilmesinde toplum-tabanlı olarak dört aşamadan oluşan çok genel bir metot geliştirmiştir.

Bu alanda çalışma yapanlardan Deprez ve Alexandre [8] çalışmalarında QSOS [9] ve OpenBRR [10] metotlarını karşılaştırarak analiz etmişlerdir. Bu çalışma, bu metotların anlaşılmasında yardımcı olmuş ama literatürdeki diğer metotlarla karşılaştırılmadığı için en uygun metotların bunlardan biri olduğu hakkında ikna edici olmamıştır.

Literatürde göze çarpan diğer bir çalışmada ise Sung, Kim ve Rhew [11] sadece kod-tabanlı değerlendirilmeye yönelik tek yönlü çalışma yapmışlar ve toplum-tabanlı değerlendirme yapmamışlardır.

Yukarıda verilen çalışmalar incelendikten sonra, bu çalışmada literatürde mevcut çalışmalardan bir sentez yapılarak kendi katkılarımızla beraber açık kaynak yazılımları hem kod-tabanlı hem de toplum-tabanlı değerlendiren kapsamlı ve iki boyutlu bir metot geliştirilmesi hedeflenmiştir. Şekil 2'de ticari yazılımların değerlendirilmesi ile ilgili

alt kısımda literatürdeki değerlendirme metotları özetlenmiş ve açık kaynak yazılımların değerlendirilmesi ile ilgili üst kısımda ise bu çalışma kapsamında geliştirilen metot özetlenmiştir.



Şekil. **2.** Yazılım (kalite) değerlendirme metotları **a)** Önerilen metot **b)** Ticari yazılımlara yönelik mevcut metotlar

Ticari yazılımlar değerlendirilirken kodun iç yapısı bilinmediğinden dolayı, dış öznitelikler (external attributes) kullanılır ve literatürde bulunan bazı karar-verme (decision-making) teknikleriyle aday yazılım ürünlerine skorlar verilip siyah-kutu değerlendirmesi (black-box evaluation) ile ihtiyaçlarımızı karşılayan en optimum yazılım seçilir [5]. Ticari yazılımların aksine açık kaynak yazılım ürünlerinde ürünün koduna erişebildiğimiz için beyaz-kutu değerlendirmesi (white-box evaluation) yapılabilir ve ürüne ait içsel öznitelikler kullanılabilir [12].

3 İki boyutlu değerlendirme metodu

Bu çalışmada Şekil 2'de gösterildiği gibi kod-tabanlı ve toplum-tabanlı olarak iki boyutlu bir değerlendirme metodu önerilmiştir. Kod-tabanlı değerlendirme yapılırken ISO 25010 modelinden seçilen – kullanıcı tarafından öncelikli görülen - içsel öznitelikler metriklerle ilişkilendirilmiş ve bu metrikler kullanılarak kodun kalite öznitelikleri (quality attributes) ölçülerek alternatifler arasından en ideali seçilmeye çalışılmıştır.

Toplum-tabanlı değerlendirme yapılırken, açık kaynak yazılımlar ile birlikte sağlanan ve süreç tarihçesini açıklayan veriler ve dokümantasyon kullanılmıştır.

Geliştirdiğimiz metot aşağıda detaylandırılan dört aşamadan oluşmaktadır.

3.1 Aday yazılım ürününün belirlenmesi

Bu aşamada ihtiyaç sahipleri öncelikle hangi çeşit yazılım ürünü için değerlendirme yapacaklarını belirlemelidirler. Bu doğrultuda piyasada var olan, gereksinimlerini karşılayacak aday yazılım ürünlerini araştırmalıdırlar.

İhtiyaç duyulan aday açık kaynak yazılım ürünlerini ararken akla ilk gelen yöntem, bir internet arama motoru (Google, Yandex, Yahoo, vb.) kullanarak arama yapıp değerlendirme yapacağımız aday ürünleri belirlemektir. Eğer arama yapacağımız ürün ismini biliyorsak ona rakip olabilecek isimleri bulmaya yönelik araştırma yapmak ta uygun olur. Eğer arama yapılacak ürün ismi tam olarak bilinmiyorsa, aday olacak ürünlerden herhangi birini gözden kaçırmamak için ürün ismi ile ilgili bütün kombinasyonlar denenmelidir. Örneğin, X formatını Y formatına dönüştüren bir yazılım ürünü için arama yaptığımızda, X2Y, XtoY gibi çeşitli kombinasyonlar denenmesi lazımdır.

Arama yaparken kullanılabilecek ikinci yöntem ise ürünlerin kodlarını ve dokümanlarını sağlayan sitelerden (Apache, SourceForge, Debian ve Savannah, vb.) aday açık kaynak yazılım ürünlerini belirlemektir. Bütün bu araştırmaları yaparken ürünlerin açık kaynak yazılım lisanslarına (General Public Licence (GPL), Library or Lessor General Public Licence (LGPL), BDS-style, vb.) sahip olduğuna dikkat etmek lazımdır [7].

3.2 Kalite özniteliklerinin belirlenmesi

Bu aşamada, belirlenen açık kaynak yazılım ürün alternatiflerinin hangi kalite özniteliklerine göre değerlendirileceği ve karşılaştırılacağı belirlenir. Geçmişten günümüze önerilen kalite modellerinden en bilinirleri McCall (1977), Boehm (1978), FURPS (1992), Dromey (1995) ve ISO 9126 (2001) modelleridir. Farklı kalite modellerinin içerdiği öznitelikler (attributes) farklılık göstermektedir. Bazıları önceki modellere yeni öznitelikler eklerken, bazıları ise öncekileri kendi modellerinden çıkarmıştır.

Biz bu çalışmamızda Tablo 1'de gösterilen ve bütün modellerde önerilen ortak öznitelikleri ölçmeye çalışacağız. Tabloda da görüldüğü gibi geçmişten günümüze bütün modellerin önerdiği ortak özniteliklerin verimlilik (efficiency), güvenilirlik (reliability), fonksiyonellik (functionality), taşınabilirlik (portability), kullanılabilirlik (usability) ve bakım yapılabilirlik (maintainability) olduğu görülmektedir [12]. Bu metodun bu ortak içsel öznitelikleri ölçmek için en güncel kalite modeli olan ISO 25010'un tanımlarını (Tablo 2) kullanması öngörülmüştür. Kullanıcı bunlardan ihtiyaç duyduklarını veya önemli gördüklerini seçer.

Tablo 1. Boehm, McCall, FURPS, ISO 9126 ve Dromey'in kalite modelleri (ortak öznitelikler)

Kalite öznitelikleri	Boehm	McCall	FURPS	ISO 9126	Dromey
Verimlilik (Efficiency)	X	X	X	X	X
Güvenilirlik (Reliability)	X	X	X	X	X
Fonksiyonellik (Functionality)			X	X	X
Bakım yapılabilirlik (Maintainability)	X	X	X	X	X
Taşınabilirlik (Portability)	X	X		X	X
Kullanılabilirlik (Usability)		X	X	X	X

3.3 Değerlendirme yapılacak metriklerin belirlenmesi

İhtiyaçlar doğrultusunda elde edilen açık kaynak yazılım alternatiflerinin nicel olarak değerlendirilmesi için uygun metriklerle ilişkilendirilmesi lazımdır. Kod-tabanlı değerlendirme yapmak için belirlenen öznitelikler direk olarak ölçülemeyeceğinden dolayı uygun metrikler belirlenir.

Tablo 2. ISO 25010 kalite öznitelikleri

Kalite öznitelikleri	İçsel öznitelikler
Fonksiyonel uygunluk	uygunluk, doğruluk, birlikte işlerlik, güvenlik, uyumluluk
Güvenilirlik	olgunluk, hata dayanıklılığı, geri kazanılabilirlik, uyumluluk
Performans verimliliği	zamana göre davranış durumu, kaynak kullanımı, uyumluluk
İşletilebilirlik	uygunluk, tanınabilirlik, kullanım kolaylığı, öğrenilebilirlik, çekicilik, teknik ulaşılabilirlik, uyumluluk
Güvenlik	gizlilik, bütünlük, inkar edilememe, hesap verebilirlik, aslına uygunluk, uyumluluk
Uyumluluk	değiştirilebilirlik, birlikte varolabilme, birlikte işlerlik, uyumluluk
Bakım yapılabilirlik	modülerlik, tekrar kullanılabilirlik, çözümlenebilirlik, değişebilirlik, değiştirilebilme kararlılığı, sınanabilirlik, uyumluluk
Aktarılabilirlik	taşınabilirlik, adapte olabilirlik, yüklenebilirlik, uyumluluk

Toplum-tabanlı değerlendirme yapabilmek için ise ürünün internet sitesinin bize sağladığı birçok tarihsel arşivlenmiş veriler (elektronik postalar, sıkça sorulan sorular, problem ve hata raporları, vb.) kullanılır. Bu verileri değerlendirebilmek için analiz aşamasında kullanılabilecek nitelikte uygun metrikler bulunmaya çalışılır. Daha sonra ilişkili metrikler yorumlanarak alternatif ürünlerden hangisinin ihtiyaçlarımıza en uygun olduğu belirlenmeye çalışılır.

3.4 Analiz etme ve seçme

Bu son aşamada kod-tabanlı değerlendirme yapmak için ihtiyaçlarımıza göre belirlenen ve nicel değerler elde etmek için metriklerle ilişkilendirilen kalite öznitelikleri analiz edilir. Metriklerden nicel değerler elde etmek için yazılım ürünün açık kaynak kodu kod analiz araçları (code analyzer tools) ile ölçülür. Elde edilen bu nicel değerler sayesinde her bir metrik ile ilişkili kalite öznitelikleri yorumlanır. Eğer belirlenen kalite öznitelikleri için ölçülen değerler bir kaç aday ürün için birbirine yakın çıkmışsa veya değerlendirmede kararsız kalınan ürünler arasında derinlemesine analiz yapılmak isteniyorsa temsili model veriler (representetive dummy data) oluşturularak ölçmek istenilen özniteliklere uygulanır ve sonuçlar analiz edilir [7]. Toplum-tabanlı değerlendirme yapmak için yazılım ürününün internet sitesindeki metriklerle ilişkilendirilen tarihsel veriler, bu metrik değerleri kullanılarak yorumlanır. Bütün bu değerlendirilmeler sonucu ihtiyaçlarımızı karşılayan en ideal ürün seçilmeye çalışılır.

4 Vaka Çalışması

4.1 Aday yazılım ürünlerinin özellikleri

Bu kısımda vaka çalışması olarak aynı amaca yönelik olan (amaç: kod derleme/build aracı) ve pratikte benzer popülerliğe sahip iki tane aday açık kaynak yazılım ürünü ele alınmış ve bu ürünler çalışmamızda önerdiğimiz metot kullanılıp değerlendirilerek bizim için en ideal olanı seçilmeye çalışılmıştır. Değerlendirme yapılacak ürünlerin (Apache Ant ve Apache Archiva) detayları Tablo 3'te gösterilmiştir.

Tablo 3. Ürünler hakkında bilgiler

Ürün özellikleri	Apache Ant	Apache Archiva
İnternet sitesi	http://ant.apache.org/	http://archiva.apache.org/
Ürün türü	Java tabanlı derleme aracı	Java tabanlı derleme aracı
Prog. dili	JAVA	JAVA
Ürün statüsü	Aktif	Aktif
İncelenen sürüm	Apache ant 1.9.7 (2016-04-12)	Apache archiva 2.2.0 (2015-03-02)
Üretim yılı	2000	2006
E. posta arşivi	http://ant.apache.org/mail.html	http://archiva.apache.org/mail-lists.html
Hata listeleri	http://issues.apache.org/bugzilla/buglist.cgi?product=Ant	http://issues.apache.org/jira/browse/MRM

4.2 Özniteliklerin ve metriklerin belirlenmesi

Aday yazılım ürünleri belirlendikten sonra, kod-tabanlı değerlendirme yapmak için öncelikle ölçülmek istenen öznitelikler belirlenmelidir. Bu vaka çalışmasında Tablo 1'de gösterilen bütün kalite modelleri tarafından önerilmiş özniteliklerden biri olan bakım yapılabilirlik (maintainability) seçilmiştir. Bu özniteliği direk olarak ölçemediğimiz için bu öznitelikle ilişkili Tablo 2'de gösterilen ve ISO 25010'nun tanımları kullanılmıştır. Daha sonra literatürde kullanılan metrikler araştırılmış ve her bir içsel özniteliğin nicel olarak ölçülmesi için Tablo 4'te gösterilen uygun metrikler bulunmuştur [13]. Bu metrikler açık kaynak yazılım ürünlerinin kaynak kodlarının sınıf (class) seviyesinde Understand Scitool adlı kod analiz aracı [14] kullanılarak ölçülmüş ve en büyük (Maks), en küçük (Min), ortalama (Ort) ve standart sapma (SS) değerleri Tablo 5'te verilmiştir.

Tablo 4. Metriklerin listesi ve kısaltmaları

Siklomatik Karmaşıklık **(CC)**	Sınıfın Ağırlıklı Metot Sayısı **(WMC)**
Kalıtım Ağacının Derinliği **(DIT)**	Sınıfın Tetiklediği Metot Sayısı **(RFC)**
Alt Sınıf Sayısı **(NOC)**	Açıklamaların Sayısı **(NOS)**
Nesne Sınıfları Arasındaki Bağımlılık **(CBO)**	Sınıf Açıklama Sıklığı **(CCF)**
Metotların Uyum Eksikliği **(LOC)**	İç İçe Döngü Sayısı **(NNL)**

Tablo 5. Kod-tabanlı değerlendirme sonuçları

		Apache ANT				Apache ARCHIVA			
Nitelik	**Metrik**	**Maks**	**Min**	**Ort**	**SS**	**Maks**	**Min**	**Ort**	**SS**
Çözümlenebilirlik (analyzability)	**CC**	51	1	2.069	2.622	52	1	1.648	2.009
	NOS	1042	0	56,309	96,384	905	0	62,067	96,004
	WMC	34	0	0,384	1,887	15	0	0,288	0,2434
	CCF	29	0	3,179	3,144	17	0	0,552	1,496
Değişebilirlik (changeability)	**NNL**	20	0	1.385	1.556	9	0	1.064	1.450
	CBO	40	0	3,541	4,749	51	0	3,633	5,998
	LOC	100	0	39,676	36,532	100	0	36,498	37,000
	DIT	7	1	1,997	1,240	5	1	1,583	0,766
Kararlılık (stability)	**NOC**	145	0	0,467	4,302	16	0	0,263	1,374
	CBO	40	0	3,541	4,749	51	0	3,633	5,998
	DIT	7	1	1,997	1,240	5	1	1,583	0,766
Sınanabilirlik (testability)	**NNL**	20	0	1,385	1,556	9	0	1,064	1,450
	RFC	663	0	36,666	34,780	112	0	26,011	18,260
	NOC	145	0	0,467	4,302	16	0	0,263	1,374

4.3 Bulgular ve Bulguların Tartışılması

Kod-tabanlı değerlendirmede elde edilen metriklere göre şu sonuçlar elde edilmiştir:

DIT metriği sınıfın kalıtım ağacının köküne uzaklığını ölçer [15]. Ağaç derinliğinin fazla olması daha fazla sınıf ve metot içereceği için karmaşıklığı arttırır ve yazılımın değişebilirliğinin (changeability) ve ürünün kararlılığının (stability) düşük olduğunun göstergesidir. Bu da istenmeyen bir durum olduğu için Tablo 5'teki sonuçlara göre bu metrik için içsel özniteliklerden değişebilirlik ve kararlılık bakımından Apache Archiva'nın daha iyi olduğunu gösterir.

WMC metriği bir sınıftaki metotların karmaşıklık derecesi ve sayısıdır [16]. Metotların sayısı arttıkça kodun çözümlenebilirlik (analyzability) süresi de otomatik olarak artacaktır. Dolayısıyla Tablo 5'teki sonuçlara göre çüzümlenebilirlik bakımından Apache Archiva yazılım ürününün daha iyi olduğunu anlaşılır.

LOC metriği metotların birbiriyle benzerlik derecesini ölçer [16]. Bu metriğin değerinin düşük olması istenir. Dolayısıyla Apache Archiva ürününün metotları diğer ürüne göre daha uyum içerisinde ve değişebilirliği daha yüksektir (Tablo 5).

RFC metriğinin değeri bir sınıftan bir nesnenin metotları çağırılması durumunda, bu nesnenin tetikleyebileceği tüm metotların sayısıdır. Yani, bir sınıfta yazılan ve çağrılan toplam metot sayısıdır [15]. RFC metriğinin değerinin düşük olduğu yazılım ürünleri daha anlaşılır ve sınanabilirdir. Bu yüzden Tablo 5'te bu metriğin değerlerine göre Apache Ant ürününün test edilmesi ve hata ayıklaması daha zordur.

NOC metriği bir sınıftan türetilmiş alt sınıfların sayısını ölçer. Bu metriğin değerinin fazla olması yeniden kullanımının yüksek olduğu, daha çok hatanın oluşabileceğini

[17] ve test yapılırken harcanan çabanın yüksek olduğunu gösterir [15]. Dolayısıyla Apache Archiva ürününün sınanabilirliği (testability) daha yüksektir.

CBO metriği sınıfın bağlı olduğu sınıf sayısını ölçer. Bu bağımlılık sınıf içerisindeki bazı özelliklerin veya metotların başka sınıflarda, sınıflar arasında kalıtım olmaksızın kullanılması durumundaki bağımlılıktır [17]. Sınıflar arasındaki bağımlılığın fazla olması modüler tasarıma zarar verir [15] ve değişebilirliği azaltır. Bu metriğin değeri bakımından arada fazla fark olmamakla birlikte Apache Ant ürünü değişebilirlik ve kararlılık bakımından biraz daha üstündür.

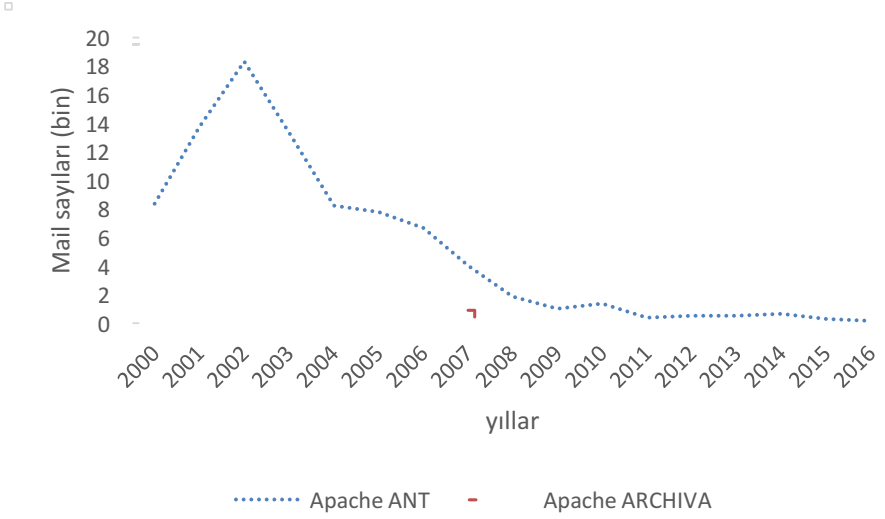
CC metriği program kaynak kodunun akışının birbirinden bağımsız yolları takip etme oranını ölçer ve direk kodun karmaşıklığı ile ilgilidir. Bu metriğin değerinin büyük olması istenmeyen bir durumdur ve kaynak kodun çözümlenebilirliğine etki eder. Dolayısıyla çözümlenebilirlik bakımından Apache Archiva ürünü diğer ürüne göre biraz daha üstündür.

NNL metriği bir sınıf içerisindeki döngülerin iç içe geçme derinliğini ölçer ve ne kadar büyük bir değer alırsa kodun sınanabilirliği ve kararlılığı azalır. Dolayısıyla bu metriğin sonuçlarına göre sınanabilirlik ve kararlılık açısından Apache Archiva ürünü diğer üründen öndedir.

NOS ve **CCF** metrikleri program içerisindeki karmaşıklığı azaltmak adına bize yol gösterecek yorumların ve açıklamaların sıklığını ölçer. Programın takip edilmesini ve çözümlenebilmesini kolaylaştırır. Tablo 5'teki sonuçlara baktığımızda NOS metriği açsından çözümlenebilirlik için Apache Archiva ürünü üstünken, CCF metriği için Apache Ant ürünü üstündür.

Sonuçlardan da görüldüğü gibi kod-tabanlı değerlendirme açısından Apache Archiva Apache Ant yazılım ürününe göre daha üstün özelliklere sahiptir. Tablo 5'te verilen ISO 25010'nun içsel öznitelikleri (çözümlenebilirlik, değişebilirlik, ...) bakım yapılabilirliğin ölçülmesini kolaylaştıran alt öznitelikler olduğu düşünüldüğünde, genel manada Apache Archiva açık kaynak yazılım ürünü ileride değişen ihtiyaç sahibi isteklerine göre yenilenebilen ve çok daha kolay adapte olabilen bir yapıya sahiptir.

Belirlenen aday yazılım ürünleri arasında toplum-tabanlı değerlendirme yapılabilmesi için yazılım ürünlerinin internet sitesindeki tarihsel verilere – elektronik posta arşivi ve problem raporlarına – erişilmiştir. Bu verilerin depolanması için açık kaynak yazılım ürününün internet sitesinde bulunan Concurrent Version Control (CVS) arşivi ve Problem Reporting Database (BUGDB) gibi veri tabanları kullanılmaktadır [18]. Bu veri tabanlarından elde ettiğimiz ürün ile ilgili tarihsel veriler, ürünün kalite özniteliklerini ölçmemizi sağlayan toplum-tabanlı değerlendirmelere katkı sağlayarak, ürün kalitesini ölçmemizde ikincil düzeyde yardımcı olmuştur.



Şekil. 3. Elektronik posta sayılarının yıllara göre değişimi

Elektronik posta verilerinin Şekil 3'te gösterildiği sonuçları şu şekilde analiz edilmiştir: Apache Ant yazılım ürünü (2000 yılı) Apache Archiva yazılım ürününe (2006 yılı) göre daha eski bir üründür. Apache Ant ürünü üretildiği ilk yılda üreticiler arasındaki ürün hakkında teknik tartışmalar ve ürün ile ilgili eksiklikler içeren mesajlaşma sayısı yıllık 8.000 civarındadır. 2002 yılına gelindiğinde ürün ile ilgili teknik sıkıntılardan dolayı bu sayı 18.000 civarına çıkmıştır. 2002 yılından sonra bu mesajlaşma trafiği azalan bir tutum sergilemiş ve 2016 yıllarına geldiğimizde 150 civarına kadar düşmüştür. Görüldüğü gibi Apache Ant ürününün standart bir ürün haline gelmesi 16 yıllık bir süreç almıştır. Şu anda bile az da olsa ürünün eksiklikleri ve teknik tartışmalar devam etmektedir.

Apache Archiva yazılım ürünü Apache Ant ürününe göre daha yeni bir ürün olmasına rağmen piyasaya sürüldüğü ilk yılda sayı olarak yıllık 900 civarında mesajlaşma trafiği ile başlarken bu sayı giderek düzenli olarak azalmış ve 2016 yılına geldiğimizde sayı olarak yıllık 30 civarına düşmüştür. Apache Ant'a göre daha kısa sürede daha az mesajlaşma trafiği ile standart bir ürün haline gelmiştir. Bu yüzden iki ürün arasındaki üreticiler arasındaki mesajlaşma trafiği dikkate alındığında Apache Archiva yazılım ürünü diğerine göre açık ara üstünlük sağlamıştır.

Tablo 6. Hata (Bug) sayılarının karşılaştırılması

	Toplam hata sayısı	**Çözülen hata sayısı**	**Başarı oranı**
Apache Ant	5.962	5.135	%86,13
Apache Archiva	1.873	1.654	%88,30

Tablo 6'da gösterilen problem raporlarının sonuçlarına göre Apache Ant yazılım ürününün toplam hata (bug) sayısının 5.962 olduğu ve bunların 5.135 tanesinin çözüldüğü görülmektedir. Bu sonuçlara göre bu ürünün hataların bulunup çözümlemesindeki

başarı oranı %86,13 olduğu görülmektedir. Apache Archiva ürününde ise toplamda 1.873 hatanın 1.654 tanesi çözülmüş ve başarı oranı Apache Ant yazılım ürününden daha yeni bir ürün olmasına rağmen %88,30 çıkmıştır. Bu oran Apache Archiva ürününün bulunan hataları çözmede daha başarılı olduğunu göstermiştir.

Tablo 7. Bulunan hataların önem derecelerine göre dağılımları

	Engelleyici (Blocker)	**Kritik** (Critical)	**Büyük** (Major)	**Küçük** (Minor)	**Önemsiz** (Trivial)
Apache Ant	%13	%14	%40	%31	%2
Apache Archiva	%1	%5	%79	%14	%1

Tablo 7'de toplam hata sayılarının önem derecelerine göre dağılımları gösterilmiştir. Bulunan bu hata çeşitlerinin ürün üzerindeki etkileri farklı olacağından bu hatalara (Engelleyici=5, Kritik= 4, Büyük= 3, Küçük=2 ve Önemsiz=1) önem sırasına göre ağırlık verilmiş ve bu hataların ürün üzerindeki etkisi ise şu şekilde hesaplanmıştır:

Apache Ant : (13x5)+(14x4)+(40x3)+(31x2)+(2x1) = 305 (1)

Apache Archiva : (1x5)+(5x4)+(79x3)+(14x2)+(1x1) = 291 (2)

Denklem 1 ve Denklem 2 incelendiğinde Apache Ant ürünü pazara çıkış süresinden itibaren karşılaştığı hataların daha ciddi hatalar olduğu anlaşılmaktadır. Bu sonucun ise ürün seçiminde tercih edilebilirliği negatif yönde etkileyebileceği düşünülmektedir.

Dolayısıyla hem kod-tabanlı değerlendirme sonuçlarına hem de toplum-tabanlı değerlendirme sonuçlarına göre iki aday yazılım arasında Apache Archiva ürününün Apache Ant ürününe göre daha tercih edilebilir ürün olduğu sonucu çıkarılmıştır.

5 Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada ihtiyaç sahiplerinin alternatif açık kaynak yazılım ürünleri arasından kendi gereksinimlerine göre değerlendirme ve seçme yaparken takip edecekleri standart bir yöntem konusunda yeterli akademik çalışma bulunmayışından yola çıkılmıştır. Bu eksikliği gidermek için literatürdeki yöntemler analiz edilmiş ve kapsamlı bir iki boyutlu yöntem geliştirilmiştir. Bu yöntemle açık kaynak yazılım ürününün hem kod-tabanlı hem de toplum-tabanlı değerlendirilmesi yapılmıştır. Bu geliştirilen yöntemi desteklemek için bir vaka çalışması yapılmış ve aynı maksada yönelik iki alternatif ürün arasından kalite gereksinimlerini daha iyi karşılayan ürün seçilmeye çalışılmıştır. Özellikle toplum tabanlı değerlendirmeye yönelik değerlendirmeler henüz başlangıç seviyesinde olmasına rağmen, tartışmaları tetikleyici ve bundan sonraki çalışmalara yol gösterici niteliktedir.

Bu geliştirdiğimiz yöntem gelecekte, kod-tabanlı değerlendirme yapmak için Tablo 1'de görülen kalite modellerinin hepsinin ortak olarak önerdiği tüm öznitelikler için uygulanmaya çalışılacaktır. Ayrıca toplum-tabanlı değerlendirme yapmak için açık kaynak yazılım ürünleri internet siteleri derinlemesine analiz edilip değerlendirme yapmak için yeni tarihsel veriler ve veri analiz yaklaşımları bulunmaya çalışılacaktır.

Kaynaklar

1. Open-source software. URL https://en.wikipedia.org/wiki/Open-source_software
2. TR Açık Kaynak Kod Platformu, URL https://acik-kaynak.org.tr
3. Noyes, K. Senior U.S. Correspondent, PCWorld, Apr 17, 2013, http://www.pcworld.com/article/2035651/open-source-is-taking-over-the-software-world-survey-says.html
4. Maki-Asiala, P., Matinlassi, M.: Quality Assurance of Open Source Components: Integrator Point of View. In: 30th Annual Int. Computer Software and Applications Conference (2006)
5. Taytaş, E.F., Gün, M., Dinçer, K., Baştüzel, S., Tekin, B.: Kamu Kurumları Tarafından Yazılım Satın Alma Sürecinde Kullanılacak Etkin Bir Yöntem Geliştirilmesi. In: Proc. of the 9th Turkish National Software Engineering Symposium. Izmir, Turkey (2015)
6. Hauge, O., Osterlie, T., Sorensen, C.F., Garea, M.: An Empirical Study on Selection of Open Source Software-Preliminary Results. In: Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE, (2009)
7. Wheeler, D.A.: How to Evaluate Open Source Software/Free Software (OSS/FS) Programs. URL http://www. dwheeler. com/oss_fs_eval. html, (2007)
8. Deprez, J.C., Alexandre, S.: Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS. In: Int. Conference on Product Focused Software Process Improvement. Springer, (2008)
9. QSOS. Method for Qualification and Selection of Open Source Software (QSOS), version 1.6. URL http://www.qsos.org.
10. Business Readiness Rating for Open Source. URL http://www.openbrr.org.
11. Sung, W.J., Kim, J.H., Rhew, S.Y.: A Quality Model for Open Source Software Selection. in Advanced Language Processing and Web Information Technology. In: Sixth International Conference on. IEEE, (2007)
12. Rawashdeh, A., Matalkah, B.: A New Software Quality Model for Evaluating COTS Components. In: Journal of Computer Science, 2(4), p. 373-381 (2006)
13. Samoladas, I., Gausios, G., Spinellis, D., Stamelos, I.: The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation. In: Open Source Development, Communities And Quality. Springer, p. 237-248 (2008)
14. Scitools | Build Notes. URL https://scitools.com/build-notes/
15. Erdemir, U., Tekin, U., Buzluca, F.: Nesne Yönetimli Yazılım Metrikleri ve Yazılım Kalitesi. Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu, İstanbul (2008)
16. Calp, M.H., Arıcı, N.: Nesne Yönelimli Tasarım Metrikleri ve Kalite Özellikleriyle İlişkisi. Politeknik Dergisi 14.1 (2011)
17. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6): p. 476-493 (1994)
18. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology. 11(3): p. 309-346 (2002)