Kamu Kurumları Tarafından Yazılım Satın Alma Sürecinde Kullanılacak Etkin Bir Yöntem Geliştirilmesi

Emine Firuze TAYTAŞ, Mehmet GÜN, Kıvanç DİNÇER,
Simge BAŞTÜZEL, Buse TEKİN

Yazılım Mühendisliği Araştırma Grubu (HUSE)
Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi, Ankara, Türkiye

kivanc.dincer@hacettepe.edu.tr

Özet. Kamu kurumları hazır ticari yazılım ürünleri satın alırken emsal ürünler arasında genelde sadece fonksiyonların ve satış fiyatının göz önüne alındığı ihalelere çıkmaktadırlar. Halbuki bir yazılımın fonksiyonel olmayan gereksinimleri, çoğu zaman fayda maliyet analizinde ve toplam edinim maliyetinde fonksiyonel gereksinimlerine baskın gelmektedir. Bu bildiride nitelikli (kaliteli) ve toplam edinim maliyeti düşük bir ürün seçilebilmesi için, satın alma sürecinde teknik puanlamaya dahil edilmesi önerilen fonksiyonel olmayan bazı özelliklere ve bunların müşteri kurum tarafından nasıl değerlendirebileceğine vurgu yapılmıştır. Mevcut literatürde bu özelliklere ilişkin değerlendirme kriterlerinin çoğunlukla geliştirici perspektifinden sunulduğu görülmektedir. Burada müşterinin kolayca vakıf olamayacağı bu tür kriterler yerine, müşteri tarafından değerlendirilebilecek kriterler tanımlanmıştır. CMMI (*Capability Maturity Model Integration*) modelinde yer alan DAR (*Decision Analysis and Resolution*) sürecini esas alan ve bahsi geçen kriterleri kullanan etkin bir değerlendirme yöntemi önerilmiştir. Yöntemin kullanımını göstermek üzere, anonim bir kamu kurumu tarafından iki farklı hazır ticari yazılım arasında bu yöntem kullanılarak yapılan seçim süreci vaka çalışması olarak anlatılmıştır.

Anahtar Kelimeler: Yazılım Mühendisliği, CMMI DAR Süreci, Yazılım Değerlendirme / Seçimi, Kamuda Yazılım Satın Alma Süreci, Vaka Çalışması

Abstract. Public institutions generally consider only the features list and the price in tenders while acquiring commercial-of-the-shelf (COTS) software products. Yet non-functional requirements of software often dominate them in terms of cost-benefit analysis and total cost of ownership (TCO). In this paper, we emphasize the non-functional requirements that need to be included in the technical scoring/evaluation in order to select a quality product with low TCO, and how the customer can evaluate those requirements. Existing literature presents the evaluation criteria for non-functional requirements from the developer perspective. The required data for such evaluation is not usually available to the customer. We here define the criteria that the customer can really use for her

evaluation and propose an effective evaluation method based on DAR (Decision Analysis and Resolution) Process in CMMI (Capability Maturity Model Integration.) To demonstrate the use of the proposed method, a case study is presented that shows the scoring of two COTS software by an anonymous public institution.

Keywords: Software Engineering, CMMI DAR Process, Software Evaluation/Selection, Public Acquisition Process, Case Study

1 Giriş

Günümüzde benzer fonksiyonlara haiz birçok alternatif hazır ticari yazılım çözümlerinin bulunması sebebiyle, özellikle kamu kurum ve kuruluşları satın alma mevzuatı gereği aradıkları fonksiyonları destekleyen en ucuz ürüne yönelmektedirler. Ancak eşdeğer gibi görünen farklı fiyat etiketine sahip yazılımlar aynı kalitede olmamakta ve çoğu zaman en ucuz alternatifin toplam edinim maliyeti uzun vadede emsallerinden daha yüksek olabilmektedir. En kaliteli ve toplam edinim maliyeti en düşük olan yazılımın seçilmesi için kullanılacak fonksiyonel olmayan gereksinimlerin değerlendirme kriterlerini ortaya koyan etkin bir yönteme ihtiyaç duyulmaktadır. Ancak mevcut literatür incelendiği zaman, bu tür kriterlerin nasıl değerlendirilebileceğini anlatan çalışmaların hep geliştirici perspektifinden yazıldığı görülmektedir. Müşterinin satın alma yaparken detaylarına vakıf olamayacağı bu tür kriterlere göre değerlendirme yapması mümkün gözükmemektedir. Bu bildiride, literatürdeki bu boşluk kısmen doldurulmaya çalışılacaktır.

Bölüm 2'de seçtiğimiz dört fonksiyonel olmayan gereksinim ve bunların müşteri perspektifinden bir bakışla değerlendirme kriterleri tanımlanacaktır. Bölüm 3'te önerdiğimiz değerlendirme yöntemi, Bölüm 4'te ise bu yöntemin kullanıldığı bir vaka çalışması anlatılacaktır. Bildiri sonuç değerlendirmesiyle son bulacaktır.

Tablo 1. Seçilen fonksiyonel olmayan özellikleri esas alan bazı yazılım kalite modelleri [11]

Fonksiyonel Olmayan Gereksinimler	McCall [17]	FURPS [18]	Dromey [19]	ISO 9126 [20]	ISO 25010 [9]
Bakım Yapılabilirlik (*Maintainability*)	X		X	X	X
Güvenilirlik (*Reliability*)	X	X	X	X	X
Kullanılabilirlik (*Usability*)	X	X	X	X	X
Performans (*Performance*)		X		X	X

2 Fonksiyonel Olmayan Özellikler ve Değerlendirme Kriterleri

Miguel ve arkadaşları [11] yazılım ürünlerinin değerlendirmesine yönelik olarak yazılım kalite modelleriyle ilgili detaylı bir sistematik literatür araştırması yapmışlar ve bu modellerde kullanılan fonksiyonel olmayan özellikleri tanımlamışlardır. Bunlar arasından yukarıda belirtilen amaçlarımıza en yakın olan ve farklı modellerde (Tablo

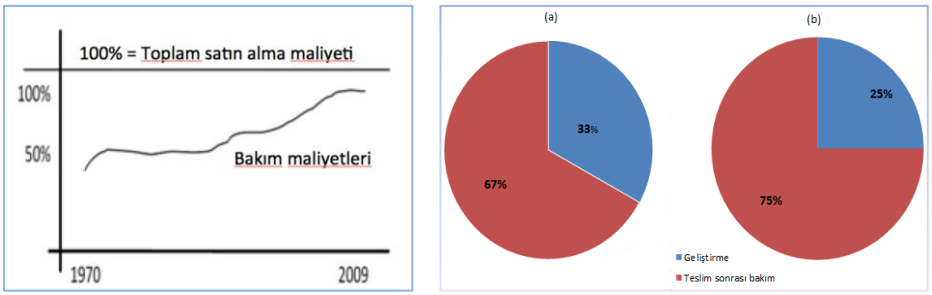
1'de modellerin bir kısmı listelenmiştir) en çok tercih edilen fonksiyonel olmayan dört adet gereksinim bu çalışmada esas alınmıştır: Bakım yapılabilirlik, güvenilirlik, kullanılabilirlik ve performans.

Müşteri kurumun belirlenen bu fonksiyonel olmayan özellikleri nasıl değerlendirmesi gerektiği konusundaki detaylar literatürdeki sınırlı sayıda yayına dayandırılarak belirlenmiş ve yazarların kendi tecrübeleriyle zenginleştirilmiştir.

Bahsi geçen fonksiyonel olmayan gereksinimler ve değerlendirme kriterleri aşağıda verilmektedir.

2.1 Bakım Yapılabilirlik (*Maintainability*)

IEEE 1219 standardına göre [12] yazılım bakımı (*software maintenance*) terimi şöyle tanımlanmaktadır: "Müşteriye teslimden sonra, hataları gidermek, performans ve diğer özellikleri iyileştirmek, ya da farklı bir ortama uyarlamak için yazılım ürününün değiştirilmesidir. "



Şekil 1. a) Bakım maliyetlerinin yıllara göre değişimi [14] b) Geliştirme ve bakım maliyetlerinin toplam maliyete oranları [13]

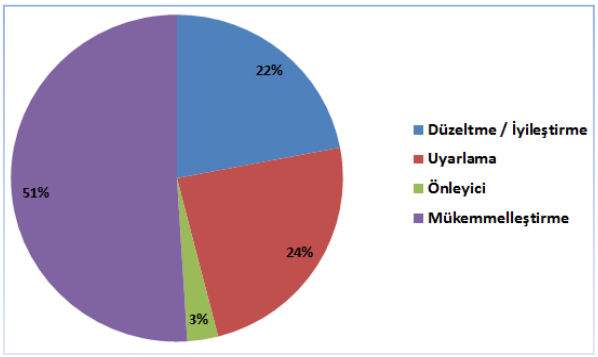
Şekil 1.a'da yıllar geçtikçe yazılımın bakım maliyetinin tüm maliyet içindeki payının arttığı görülmektedir. Sağda ise 1976-1981 ve 1992-98 yılları arasındaki geliştirme maliyetleri ve teslim sonrasındaki bakım maliyetleri karşılaştırılmıştır. Buna göre yazılımın bakım maliyeti geliştirme maliyetinin 2-3 katına ulaşabilmektedir. Bu da en baştaki tezimizi, yani yazılım satın alırken sadece satın alma bedelinin esas alınamayacağını ve fonksiyonel olmayan bakım yapılabilirlik hususunun da dikkate alınması gerektiğini teyit etmektedir.

Şekil 2'te görüldüğü üzere [10], bakım çalışmaları için harcanan zamanın

- %22'sini düzeltme/iyileştirmeler – hataların giderilmesi maksadıyla yapılan çalışmalar,
- %24'ünü uyarlamalar – ortamdaki değişiklik ve yeniliklere uyum sağlanması maksadıyla yapılan çalışmalar,
- %3'ünü önleyici bakım – yazılımın zamanla kalitesinin düşmesi karşısında yapılan ve yazılımın kalitesini arttırmaya yönelik çalışmalar,

- %51'ini mükemmelleştirmeler – yazılıma yeni davranışlar ve özellikler kazandırılmasına yönelik çalışmalar oluşturmaktadır ki;

bunlardan düzeltme/iyileştirmeler haricinde diğerleri normal garanti kapsamında ücretsiz yapılan faaliyetler değildir toplam ve satın alma maliyetine etki ederler.



Şekil 2. Bakım çeşitleri için harcanan zaman oranları [10]

Bir yazılımın bakım yapılabilir olup olmadığını değerlendirmek için, yazılımın geliştiricileri tarafından aşağıdaki kriterler kullanılabilir [2] :

- Yazılım paketini geliştirmede kullanılan kod kesimleri başka bir geliştirici tarafından anlaşılabilecek nitelikte midir?
- Yazılım paketi kaç adet modülden oluşmaktadır?
- Birbirinden bağımsız şekilde kurulum yapılabilecek kaç adet modül mevcuttur?
- Birbirinden bağımsız kod parçalarının ortalama uzunluğu (*size*) nedir?
- Farklı sunucular üzerinde dağıtılmış modüller mevcut mudur?
- Yazılım paketi geliştirilirken kullanılan platform ve programlama dili nedir?
- Ürün çalışırken değiştirilebilir (*swappable*) bileşenlere sahip mi? (Örneğin sistem bakıma girdiği takdirde kullanıcı programa erişebiliyor mu, bilgilerini görüntüleyebiliyor mu?)

Müşteri açısından ise, yukarıdaki kriterler vakıf olunabilir ve kontrol edilebilir değildir. Müşteri kurum ancak aşağıdaki kriterlere bakarak ürünün bakım yapılabilirlik özellikleri değerlendirebilir:

- Ürün kullanımı sırasında teknik bir problem olduğu zaman teknik destek sağlanıyor mu? Sağlanıyorsa hangi şartlarda sağlanıyor? Örneğin şirkette veya müşterinin yerinde, ücretli/ücretsiz, telefonla destek var mı?
- Ürün muhtemel platform değişikliklerine uyum sağlar mı?
- Ürün değişen ortam koşullarına göre güncelleme desteği sağlıyor mu? Güncellemeler hangi sıklıkta yapılıyor?
- Destek portali var mı? Güncelleme tarihçesi vs. ilişkin portalde bilgi var mı? Güncellemeler profesyonel destek gerektiriyor mu?
- Ürün kurulum ve yeniden kurulum desteği var mı? Bu destek nasıl ve hangi şartlarda sağlanıyor?

- Sistem üzerinde değişiklikler (bakım vb.) yapılırken sistemin çalışması bu durumdan etkileniyor mu?

Yukarıdaki hususlar müşteri kurum tarafından ürünle ilgili fizibilite çalışması yapılırken dikkate alınmalıdır. Ürün alınırken de şartnamelere / sözleşmelere bu konuda değerlendirme yapabilmek için gerekli maddeler konulmalıdır.

2.2 Güvenilirlik (*Reliability*)

IEEE'ye [3] göre güvenilirlik; bir bileşenin veya yazılımın belirli koşullar altında belirli zaman süreci içinde gerekli fonksiyonelliklerini ne kadar karşıladığıdır. Güvenilirlik kavramının önemini, örnek olarak bir yolcu uçağının yazılımında gerçekleşen çökme durumunun yol açabileceği olumsuz şartları düşünerek tahayyül edebiliriz.

Ancak güvenilirliği değerlendirmek kolay bir iş değildir. Bazen yazılımların web siteleri güvenilirlik hakkında ipucu verebilmektedir. Web sitesinde verilen yazılımın geliştirilme ortamı ve durumu, yazılımın son kullanıcıların kullanımına hazır olup olmadığı gibi bilgiler alıcıya yazılımın güvenilirliğini ölçmede kısmen yol gösterebilir [7]. Aynı zamanda bir yazılımın güvenilirliği, üretimi sırasında ne kadar doğru ve yeterli test edildiği ile de direkt olarak alakalıdır.

Güvenilirlik yazılımın nasıl kullanıldığıyla sıkı sıkıya ilişkilidir dolayısıyla güvenilirliği ölçmenin en etkin yolu onu gerçek kullanım ortamında kullanarak test etmektir [7]. Müşteri kendi ortamında yazılımı kurup belli bir süre gerçek verilerle, mevcut işleyişine paralel olarak; pilot uygulama yapmalıdır.

Örneğin bir aylık bir pilot uygulama sonrasında temel olarak aşağıdaki sorular öne çıkmaktadır [3]:

- Yazılım tutarlı ve hatasız bir şekilde çalışabiliyor mu? Bir ay içerisinde yazılımda kaç adet hata yakalandı?
- Yazılım kendi içinde hata önleme ve çökme engelleme mekanizmalarına sahip mi? Bir ay içerisinde yazılımda kaç kere çökme oldu? Çökme durumlarının ne kadarında kurtarma işlemi sağlandı? Yazılım ne kadar sıklıkta çökme ve kayıplara sebep olmakta?
- Yazılım kendi içinde yedekleme ve geri yükleme mekanizmalarına sahip mi?

Bazen de pilot uygulama yerine benzer bir başka bir müşterinin tecrübeleri değerlendirme yapmak için kullanılabilir.

2.3 Performans (*Performance*)

Performans tasarım, kod ve çalıştırma ortamı gibi her türlü durumdan etkilenebilmektedir [6]. Performansı iyi düzeyde olmayan bir yazılıma yapılacak yatırım müşteri için kayıp olacaktır [5]. Sorgu tabanlı çalışan bir sistemde, kullanıcıların sorgu isteklerine çok geç dönebilen bir yazılım, kullanıcıları tatmin etmeyecektir. Bu sistemin bir bankacılık sistemi olduğu düşünülürse bu hususun önemi daha belirgin hale gelmektedir.

Bir yazılımın performansını değerlendirmede, müşteri perspektifinden bakıldığında, temel olarak şu kriterler öne çıkabilir:

- Sorgularda sistemin cevap süresi istenen seviyede mi? [5]
- Ortalama tepki zamanı beklentileri karşılıyor mu?
- Sistem en fazla kaç kullanıcıya hizmet verebilmektedir?
- Yazılımın gerektirdiği en az donanım gereksinimleri nelerdir? (Bunların maliyeti, yazılımın edinim maliyeti hesaplanırken göz önüne alınmalıdır.)

2.4 Kullanılabilirlik (*Usability*)

Kullanılabilirlik bir yazılımın tercih edilmesinde başta gelen kalite nitelikleri arasında yer almaktadır. Bu nedenle kullanılabilirliğin doğru ve tutarlı ölçümü gerçekleştirilmelidir. Kullanılabilirlik kavramının derecesi kullanıcıdan kullanıcıya değişebilmektedir. Kullanılabilirlik şu bileşenlerin bir araya gelmesiyle oluşmaktadır:

- Öğrenilebilirlik (*Learnability*): İlk defa karşılaşılan bir programın ne kadar kolay anlaşılabildiğidir.
- Etkinlik (*Efficiency*): Anlaşılan bir programda verilen görevlerin ne kadar hızlı yapılabildiğidir.
- Hatırlanabilirlik (*Memorability*): Belirli süre kullanılmayan programlara geri dönüş yapıldığında ne düzeyde yeterli olduğudur.

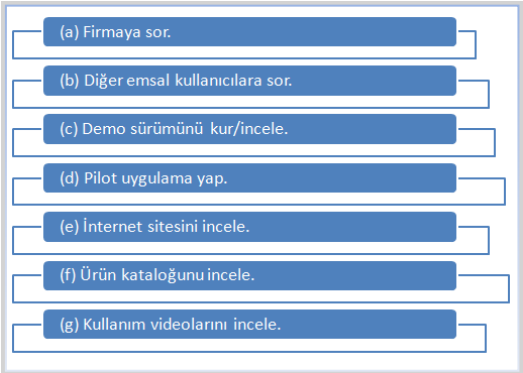
Buna göre bir yazılımın kullanılabilirliğini ölçmek için şu soruları yöneltebiliriz [15]:

- Yazılımın kullanımı sırasında kullanıcının yardıma ihtiyacı var mı?
- Bu yazılımın kullanılabilmesi için kullanıcıların belirli bir tecrübeye sahip olması gerekiyor mu? Gerekiyorsa ne kadar tecrübeli olmalıdır?
- Ürünün anlaşılmasını kolay kılacak bir dokümantasyon mevcut mu? Dokümantasyondaki bilgiler ne kadar anlaşılır? Dokümantasyon olmasa ürün kullanılabilir mi?
- Daha etkin kullanım için tasarımda değişiklik yapılabiliyor mu?
- Hata raporları kullanıcı tarafından kolaylıkla anlaşılabiliyor mu? Meydana gelen hatalardan hızlı ve kolay bir şekilde kurtulmak mümkün mü?
- Ürün kimlere hitap ediyor, hangi kesimden insanlar kullanıcı profilindedir?
- Ürün tecrübesiz veya tecrübesi az olan kullanıcı tarafından da benimsenecek şekilde geliştirilmiş midir?
- Ürün kullanıcı dostu mu? (Kullanıcı ara yüzündeki öğeler anlaşılır mı?)
- Ürün bir işlemi gerçekleştirmek için en az sayıda adımı içeriyor mu?
- Ürünün kullanımı belli bir süre geçtikten sonra hatırlanıyor mu?

Kullanılabilirlik her müşteri için farklı olacağından ürünün kullanılabilirlik açısından puanlandırılması farklı gruplara göre ayrı ayrı gerçekleştirilebilir.

3 Değerlendirme Yöntemi

CMMI'da [1] tanımlanan DAR sürecinin amacı, alternatifleri sistemli bir değerlendirme işlemine tabi tutarak en uygun olanını seçmektir. Önerilen yöntem temel olarak müşteri kurumun alternatif yazılımlardan birini seçmek için DAR sürecini kullanmasını kolaylaştırmayı ve bunu teknik puanlamaya dahil etmesini hedeflemektedir.



Şekil 3. Değerlendirmede kullanılacak bilgi edinme yöntemleri

DAR sürecini esas alan yöntemin adımları [1] aşağıda verilmiştir:

1. Alternatifleri değerlendirmek için kriterlerin tanımlanması: Kaliteli ve toplam edinim maliyeti düşük olan ürünü seçmek için Bölüm 2'de anlatılan şekilde dört fonksiyonel olmayan özellik ve ağırlıkları belirlenmiştir. Bunların değerlendirme kriterleri ise Bölüm 3'te tanımlandığı gibi belirlenmiştir. Kriterlerin hepsi aynı varsayılan ağırlığa sahiptir.
2. Alternatiflerin belirlenmesi: Değerlendirilecek yazılım ürün alternatifleri müşteri kurum tarafından seçilir.
3. Değerlendirme bilgi edinme yöntem(ler)inin seçilmesi: Her bir kriter değerlendirilirken kullanılabilecek bilgi edinme yöntemleri Şekil 3'te verilmiştir. Bunlardan uygun olanlar müşteri kurum tarafından seçilir.
4. Alternatiflerin puanlanması: Alternatiflere müşteri kurum tarafından, kurumun ihtiyaç ve beklentileri göz önüne alınarak, 1 ile 5 arası puanlar verilir.
5. Seçim kararının verilmesi: Sonuçta en yüksek puan alan alternatifin kurumun ihtiyacını en iyi şekilde karşılayan olması ve tercih edilmesi beklenir.

Önerilen yöntem temel olarak bu çalışma kapsamında geliştirilen ve Tablo 2'de verilen puanlama tablosunun bir kısmının hazır olarak sunulmasına diğer kısımlarının da müşteri kurum tarafından uygun şekilde doldurulması için rehberlik sağlanmasına dayanmaktadır. Bu tablo Özellikler ve Değerlendirme Kriterleri, Beklentiler,

Değerlendirme Yöntemleri, Ağırlık, Puan ve Toplam Puan sütunlarından oluşmaktadır.

Özellikler ve Değerlendirme Kriterleri sütununda süreçte kullanacağımız dört adet fonksiyonel olmayan özellik ve bunların her birinin müşteri kurum tarafından değerlendirmesine yardımcı olacak kriterler listelenmiştir. Her bir özellik aynı zamanda bir sonraki sütunda yer alan kriterleri gruplandırmaktadır.

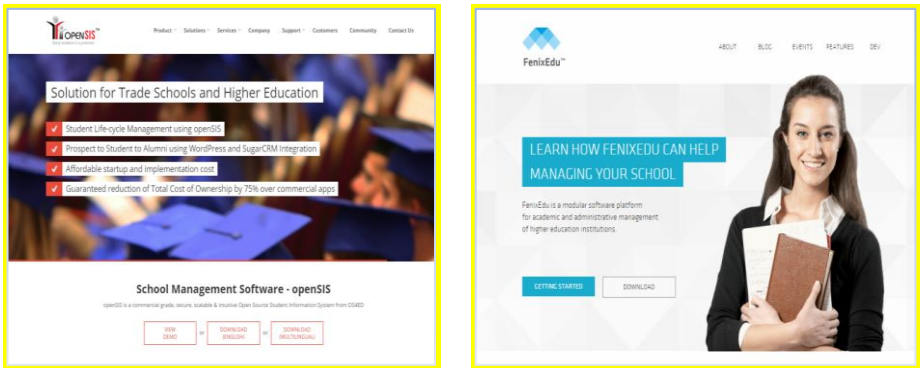
Beklentiler sütunu, müşterinin kontrol listesindeki sorulara karşılık asgari veya azami düzeyde beklentilerini belirtebileceği alanı belirtmektedir. Mesela "Sistem en fazla kaç kullanıcıya hizmet verebilmektedir? " sorusuna karşılık beklenti bölümüne "asgari 500, azami 1500" gibi bir ifade yazılabilir. Bu ifade ilgili sorunun sonraki aşamada 1 (en kötü) – 5 (en iyi) olarak puanlanmasında yol gösterici olacaktır.

Değerlendirme (bilgi edinme) yöntemleri sütunu müşteri kurumun ilgili kriter hakkında bilgi edinmek istediği zaman kullanabileceği yöntemlerden (Şekil 3) bir veya daha fazlasını sırasıyla seçmesini gerektirir.

Ağırlık sütununda ise puanlamada kullanılacak olan ve fonksiyonel olmayan özelliklere göre belirlenen katsayılar yazarlar tarafından önceden belirlenmiş olarak verilmektedir. Bu ağırlıklar müşterinin ürün seçiminde hangi kriterlerin daha ağır bastığı gösterir. Bu katsayılar belirlenirken toplam satın alma maliyetinin düşük olması amacıyla bakım yapılabilirlik özelliğine 0,60/1,00; ürün kalitesi grubundaki özelliklere ise toplamda 0,40/1,00 ağırlık verilmiştir. Bu ikinci gruptaki özelliklere yazılım kalite modellerindeki kullanımları [11] göz önünde bulundurularak alt ağırlıklar verilmiştir. Müşteri kurumun biraz tecrübe kazandıktan sonra bu ağırlıkları satın alma hedeflerine göre her bir seferinde değiştirmesi mümkündür.

Puan sütununda müşteri kurumun kriterlere göre beklentilerini de göz önüne alarak "Alternatif A" ve "Alternatif B" yazılımlarını puanlaması beklenmektedir. Örneğin "Yazılımın dokümantasyon desteği var mı?" sorusuna varsa 4 veya 5 yoksa 1 şeklinde puan verilebilir. Dokümanın kalitesi puanı belirler. Eğer bir kriter değerlendirme için önemli görülmüyorsa, buna puan verilmez.

Ağırlıklı Puan sütunu ise kriterler için ayrı ağırlık verildiyse bunun sonucunu gösterir, aksi halde puanın 5'e bölümüyle elde edilir. Kriterlere verilen ağırlıklı puanlar toplandıktan sonra bu toplam ilgili özelliğin ağırlık sütununda yer alan katsayı ile çarpılarak "Kategori Toplam Puanı" bulunur. Bunların toplamı ise ürünün toplam değerlendirme puanını verir.



Şekil 4. Değerlendirilecek ürünlerin web sitelerinden birer kesit

4 Vaka Çalışması

Bu kısımda vaka çalışması olarak anonim bir kamu kurumu tarafından sertifika programlarıyla ilgili öğrenci işleri otomasyonu için kullanılabilecek hazır ticari yazılımların değerlendirilmesinde önerilen yöntemin nasıl kullanıldığı anlatılacaktır.

Değerlendirilen yazılımlardan birincisi Opensis [4] ikincisi ise Fenix [16] isimli ticari ürünlerdir. İlk incelemede fonksiyonlarına dayalı olarak her iki ürünün de kurumun ihtiyacını karşılayabileceği değerlendirilmiştir, ilk aşama değerlendirme ürünlerin web sitelerinde (Şekil 4) yer alan bilgilerden faydalanılarak yapılmıştır.

Yöntemin nasıl çalıştığını açıklamak için değerlendirme tablosunun web sitesinde bulunan bilgiler yorumlandıktan sonraki durumu Tablo 2'de verilmiştir. Her kriterle ilgili puanlamaya esas olan gerekçeler kontrol edilebilirliğini ve objektifliği sağlamak açısından tabloyla birlikte ayrıca verilmelidir. Burada bazı değerlendirmelerimizi özet olarak vereceğiz.

Bakım yapılabilirlik kategorisinde;

- Opensis ürününün ücretsiz destek portali olmasına rağmen Fenix'te böyle bir portal yoktur. Opensis 5, Fenix 1 puan almıştır. (1. ve 2. soru)
- Opensis ürünün Agile yöntemle geliştirilmesi iyileştirmelerin sıklıkla yapıldığına işarettir. Fenix'te ise güncelleştirme durumu hakkında detaylı bilgi bulunmamaktadır. Opensis 5, Fenix 3 puan almıştır. (4. soru)
- Opensis ürününün son versiyonu ve yayınlanma tarihi sitede verilmiştir. Güncellemelerin tarihçesi ile ilgili olarak Fenix'te detaylı bilgi olmadığından Opensis 5, Fenix 1 puan almıştır. (6. soru)
- Ürünlerle ilgili bazı istisnalar da mevcuttur. Mesela Opensis yazılımı cloud ortamına yedek alabilmekte iken Fenix bunu yapamamaktadır. Bu özellik beklentiler arasında belirtilmediği için her iki yazılım da bu kriterden puan almamıştır. (10. soru)

Güvenilirlik kategorisinde;

- Her iki ürün de online olarak demo modunda çalışma imkanı vermektedir. Dolayısıyla 5 puan verilmiştir. (11. Soru)
- İki yazılımda da hata engelleme ve çökme mekanizmaları ile ilgili bilgi bulunmadığından ve iki yazılımdan da bu mekanizmaların bulunması beklendiğinden ilgili soruya iki yazılım için de 1 puan verilmiştir. (12.soru)
- Ürünün kullanıcıları *community* sekmesinde listelenmektedir. Böylece değerlendirme yönteminde yer alan "başka kullanıcılara sor" seçeneğine, müşteri kolayca başvurabilecektir.

Performans kategorisinde;

- Yapılan pilot uygulamalarda sistemin sorgu süreleri beklentiye göre kıyaslanmıştır. Fenix daha tatmin edici bir sürede çalışabildiğinden Opensis'e 3, Fenix'e 4 puan verilmiştir. (14. soru)
- Yine aynı pilot uygulamalarda yazılımların girdilere karşı verdiği tepki süreleri beklentiye aynı seviyede yakın olduğundan iki yazılıma da tepki süreleri ile ilgili soru için 4 puan verilmiştir. (15. soru)

Tablo 2. Alternatif Ürün Değerlendirme Tablosu

					A. OPENSIS	B: FENIX		
FONKSİYONEL OLMAYAN ÖZELLİK KATEGORİLERİ	KRİTERLER	BEKLENTİLER	DEĞERLENDİRME YÖNTEMİ	AĞIRLIK	PUAN (1 - 5)		AĞIRLIKLI PUAN (Derece x Ağırlık)	
					A	B	A	B
Bakım Yapılabilirlik (Maintainability)	1- Ürün kullanımı sırasında teknik bir problem olduğu zaman teknik destek sağlanıyor mu? Sağlanıyorsa hangi şartlarda sağlanıyor? Örneğin şirkette veya müşterinin yerinde, ücretli/ücretsiz, telefonla destek var mı?	Pzt-Cuma mesai saatlerinde telefonla ücretsiz destek	e, a		5	1	1,0	0,2
	2- Sıkça sorulan soruların ve forumların yer aldığı destek portali var mı?	Olmalı	e, a, g		5	1	1,0	0,2
	3- Ürün muhtemel platform değişikliklerine uyum sağlar mı?	Windows ve iOS	e, a, b		5	3	1,0	0,6
	4- Ürün değişen ortam koşullarına göre güncelleme desteği sağlıyor mu?	O/S güncellemeleri	a		5	3	1,0	0,6
	5- Güncellemeler hangi sıklıkta yapılıyor?	Senede iki ana sürüm	e, a, b		1	1	0,2	0,2
	6- Güncelleme tarihçesi vs. ilişkin portalda bilgi var mı?	Olmalı	e		5	1	1,0	0,2
	7- Güncellemeler profesyonel destek gerektiriyor mu?	Gerektirmemeli	a, b		1	1	0,2	0,2
	8- Ürün kurulum ve yeniden kurulum desteği var mı?	Fark etmez	e, a		5	5	1,0	1,0
	9- Online olarak güncel dokümantasyona ulaşılabiliyor mu ?	Ulaşılabilmeli	e, f, g		5	5	1,0	1,0
	10- Ürün cloud üzerinden paylaşımlı hosting servisiyle entegre olabiliyor mu?	Olmalı	e, f, a		-	-	-	-
		PUANI :		0,6			4,44	2,52
Güvenilirlik (Reliability)	11- Yazılımın demo sürümü ortamda tutarlı ve hatasız bir şekilde çalışabiliyor mu?	Çalışmalı	d, b		5	5	1,0	1,0
	12- Yazılım kendi içinde hata önleme ve çökme engelleme mekanizmalarına sahip mi?	Sahip olmalı	d, f		1	1	0,2	0,2
	13- Yazılım kendi içinde yedekleme ve geri yükleme mekanizmalarına sahip mi?	Önemli değil	d, f		-	-	-	-
		KATEGORİ TOPLAM PUANI :		0,2			0,24	0,24
Performans (Performance)	14- Sistemin sorgularda cevap süresi istenen seviyede mi?	3 sn / sorgu	d, f, b		3	4	0,6	0,8
	15- Ortalama tepki zamanı müşteri beklentisini karşılıyor mu?	1 sn/ aksiyon	d, f, b		4	4	0,8	0,8
	16- Sistem en fazla kaç kullanıcıya hizmet verebilmektedir?	500 - 1500	f, e, a		4	4	0,8	0,8
	17- Yazılımın gerektirdiği en az donanım gereksinimleri nelerdir?	zami 20,000 TL'lik sunuc	f, a		1	3	0,2	0,6
		KATEGORİ TOPLAM PUANI :		0,1			0,24	0,30
Kullanılabilirlik (Usability)	18- Ürün kurum içi kullanıma ne kadar uygun?				5	5	1,0	1,0
	19- Ürün tecrübesiz veya tecrübesi az olan kullanıcı tarafından da benimsenecek şekilde geliştirilmiş midir?	Evet	f, d		5	3	1,0	0,6
	20- Bu yazılımın kullanılabilmesi için kullanıcıların belirli bir tecrübeye sahip olması gerekiyor mu? Gerekiyorsa ne kadar zamanlık bir tecrübe şartı koyulmuştur?	Gerekmemeli	f, d, e		5	3	1,0	0,6
	21- Üründe çoklu dil seçeneği mevcut mu?	Olmalı	f, e		5	5	1,0	1,0
	22- Ürünün anlaşılmasını kolay kılacak bir dökümantasyon mevcut mu?	Mevcut olmalı	e, c, a		5	5	1,0	1,0
		KATEGORİ TOPLAM PUANI :		0,1			0,50	0,42
						TOPLAM PUAN	5,42	3,48

- Fenix'in veritabanı için, sunucu üzerinde asgari 20 GB yer ayrılması gerektiği bilgisi verilmiştir. Diğer hususlarda bilgi yeterli değildir. Opensis için ise bu konuda bilgi edinilememiştir. Dolayısıyla Opensis'e 1, Felix'e 3 puan verilmiştir. (17. soru)
- Fenix'in günlük desteklediği ortalama işlem sayısının 731.296 işlem/gün olduğu belirtilmiştir. Ancak bu konuda bir beklenti belirtilmediğinden puan verilmemiştir.

Kullanılabilirlik kategorisinde;

- Fenix ürününün sitesi incelendiğinde daha çok yazılım geliştiriciler için bir *framework* niteliğinde olduğu yani Fenix'in programlama tecrübesi gerektireceği gözlemlenmiştir. Bu yüzden ürünün kullanımı programlama tecrübesi olmayan biri için kolay değildir. Opensis'te ise tecrübesiz kullanıcılar daha uygun bir ortam sunmaktadır. (19. soru)
- İki ürünün de çoklu dil desteği mevcuttur. Tam puan verilmiştir. (21. soru)
- İki ürünün de web siteleri incelendiğinde ürün kullanımını anlaşılabilir ve kolay kılacak yeterli dokümantasyon mevcuttur. (22. soru)

Geliştirilen yöntemle yazılımlara müşteri kurum tarafından verilen puanlardan elde edilen sonuçlar değerlendirilerek yazılımlar karşılaştırılmıştır. Bu aşamada Opensis ürününün daha yüksek puan alması sebebiyle müşteri beklentilerini daha iyi karşılayacağı anlaşılmıştır. Bu değerlendirme iteratif olarak farklı bilgi kaynaklarından edinilen bilgilerin tabloya yansıtılması suretiyle devam ettirilebilir. Her aşamanın sonuçlarının izlenebilir olması seçim sürecine önemli bir katkı sağlar. Satın alma sürecinde ise bu tablo seçilen fonksiyonel olmayan özelliklerin teknik puanlamasına esas olarak kabul edilecek ve istekli firmaların cevaplarında bu kriterleri adreslemesi istenecektir.

5 Sonuç

Literatürde bir yazılımın fonksiyonel olmayan özelliklerine karşılık gelen kriterlerin hep geliştirici perspektifiyle açıklandığı anlaşılmıştır. Bu çalışmada müşteri açısından değerlendirme yapılmasını sağlayan hazır ticari yazılım seçme işini kolaylaştıracak DAR tabanlı bir yöntem tanımlanmıştır. Müşteri kuruma bir yazılım ürününü değerlendirirken kolayca kullanabileceği bu yöntem kriterlerden oluşan kontrol listelerini, değerlendirme yöntemlerini ve puanlama usullerini de içermektedir. Önerilen yöntem örnek iki yazılımın karşılaştırılması için kullanılmış ve olumlu sonuçlar verdiği görülmüştür. Gelecekte daha da geliştirilmesi için çalışma yapılması planlanmaktadır.

Kaynaklar

[1] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Addison-Wesley Professional; SEI Series in Software Engineering; 3rd edition, 2011.

[2] Crouch, Steve (2015). Developing Maintainable Software, *URL: http://software.ac.uk/resources/guides/developing-maintainable-software*

[3] Dr. Rosenberg, L., Hammer, T., & Shaw, J. (1998). Software metrics and reliability. In *Proceedings of the 9th International Symposium on Software Reliability Engineering* (pp. 1–8).

[4] OPENSIS, School Management Software, *URL*: *http://opensis.com/*

[5] Williams, L. G., Ph, D., & Smith, C. U. (1998). Performance evaluation of software architectures. *Proceedings of the 1st International Workshop on Software and Performance*, (303), 164–177. doi:10.1145/287318.287353

[6] Woodside, M., Franks, G., & Petriu, D. C. (2007). The Future of Software Performance Engineering. In *Future of Software Engineering (FOSE '07)* (pp. 171–187). doi:10.1109/FOSE.2007.32

[7] Wheeler, D. (2005). How to evaluate open source software/free software (OSS/FS) programs. *URL: Http://www. Dwheeler. Com/oss_fs_eval. Html*, 1–24. Retrieved from *http://www.dwheeler.com/oss_fs_eval.html*

[8] Lyu, M. R. (1998). Design, testing, and evaluation techniques for software reliability engineering. In *Euromicro Conference, 1998. Proceedings. 24th* (Vol. 2, p. XXXIX – XXLVI vol.2). doi:10.1109/EURMIC.1998.708059

[9] ISO/ IEC CD 25010. (2008). Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) Quality Model and guide. *International Organization for Standardization*, Geneva, Switzerland.

[10] What is System Maintenance?, frontsources.com, FrontSources Information Tech, *URL: http://www.frontsources.com/sysMain.html*

[11] J. P. Miguel, D. Mauricio, and G. Rodríguez, "A Review of Software Quality Models for the Evaluation of Software Products," *Int. J. Softw. Eng. Appl.*, vol. 5, no. 6, pp. 31–53, 2014.

[12] Wikipedia, IEEE 1219, *URL: https://en.wikipedia.org/wiki/IEEE_1219*

[13] Object Oriented And Classical Software Engineering 8th Edition, Stephen R. Schach, Part 1.3.2 The Importance of Postdelivery Maintenance, pp. 11

[14] Models, S. L. (2010). Object-Oriented and Classical Software Engineering Lifecycle. *Development*, 1–38. doi:10.1036/0072554509

[15] Schneider, Daniel K. (2011), Usability and User Experience Surveys, *URL: http://edutechwiki.unige.ch/en/Usability_and_user_experience_surveys*

[16] FENIXEDU, What is Fenixedu?, *URL: http://fenixedu.org/*

[17] Dubey, S.K & Soumi Ghosh & Ajay Rana. (2012). "Comparison of Software Quality Models: An Analytical Approach," *International Journal of Emerging Technology and Advanced Engineering*, Volume 2, Issue 2, pp 111-119

[18] Grady, R. B. (1992). *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, Englewood Cliffs, NJ, USA

[19] Dromey, R. G. (1995). "A model for software product quality," *IEEE Transactions on Software Engineering*, 21:146-162

[20] ISO/IEC IS 9126-1. (2001). Software Engineering - Product Quality – Part 1: Quality Model. *International Organization for Standardization*, Geneva, Switzerland.