Form Tabanlı Uygulamalar İçin Çaba Kestirimi

Uğur Kemal Haşlak Yunus Emre Selçuk[1]

[1]Yıldız Teknik Üniversitesi, Bilgisayar Müh., 34349 Beşiktaş, İstanbul
[1] yunus@ce.yildiz.edu.tr

Özet. Yazılım geliştirme maliyetleri, ölçüm ve kestirim yöntemlerinin çeşitli yetersizliklerinden dolayı sık sık kontrol dışına çıkmaktadır. Planlanan zaman ve bütçeyi aşan çok sayıda proje mevcuttur. Zaman, gereken çaba, dolayısıyla da bütçe tahminini başta doğru yapamamak, bunun en temel nedenlerinden biridir.

Bu çalışma ile sektörde yaygın olarak kullanılan nesne yönelimli programlama dilleri ile form tabanlı uygulamaların gerçeklendiği projeler için kullanılabilecek bir kestirim modeli oluşturulması amaçlanmıştır. Bu kestirim modelinde en çok kullanılan COCOMO II.2000 modeli referans alınmıştır. Nesnelerin karmaşıklık durumu da göz önünde bulundurularak nesne / satır sayısı dönüşüm tablosundan faydalanılarak bir kestirim yapılmaya çalışılmıştır. Bu sayede pratikte uygulanabilir bir yöntem ortaya çıkmıştır. Yöntem belirli bir proje grubu için uygulanmış ve tatmin edici sonuçlar elde edilmiştir.

Anahtar Kelimeler: Yazılım çaba kestirimi, form tabanlı yazılım geliştirme, COCOMO II

1 Giriş

Yazılım geliştirme maliyetleri, ölçüm ve kestirim yöntemlerinin çeşitli yetersizliklerinden dolayı sık sık kontrol dışına çıkmaktadır. Maliyet kestirimi konusunda yapılan çok sayıda çalışmaya rağmen henüz bu problem nihai olarak çözülememiştir.

Sektörde kullanılan ilk yazılım ölçüm yöntemlerinden biri satır sayısını sayma yöntemidir. Ancak bu yöntem çok sayıda teknik sorun içermekte ve istismar edilebilmektedir. Bu nedenle daha doğru ölçüm metotları araştırılmaya başlanmıştır.

70'lerin ortasında geliştirilen çevrimsellik karmaşıklığı (Cyclomatic Complexity) [1] temel olarak yazılım karmaşıklığını ölçmeye yarayan bir graf teoreminin kullanımıdır. Çevrimsellik sayısı, minimum yol sayısının bulunması işidir. 1977'de yazılım ölçütleri konusunda yazılan ilk kitap olan "Software Metrics" adındaki kitap yayınlanmıştır [2]. 1978 yılında, yazılım geliştirme yaşam döngüsü süreçlerine göre değişkenlik gösteren çaba ve maliyet ihtiyaçlarını dikkate alan Putnam modeli [3] ortaya konmuştur. 1979'da işlev puanın (FP: Function Point) tanımlanması önemli bir adımdır [4]. 1981 yılında Boehm, COCOMO modelini tanıtmıştır [5]. Londeix 1987 yılında yaptığı çalışmada makro ve mikro kestirim tekniklerinin olduğunu belirtmiştir

[6]. 1995 yılında COCOMO II tanıtılmıştır [7]. 2000 yılında Boehm ve arkadaşları tarafından yapılan bir araştırmada mevcut kestirim yöntemleri sınıflandırılmıştır [8].

Jorgensen ve Shepperd 2007 yılında yazılım maliyet kestirimi alanında o zamana kadar yapılan akademik çalışmaları incelemiştir [9]. Bu çalışmada 76 ayrı dergide yer alan 304 makale incelenmiştir. Bu çalışmanın sonucuna göre regresyon tabanlı yaklaşımların tüm çalışmalar içindeki payı %49 dur. Bu da regresyon tabanlı yöntemlere ilginin diğerlerine nazaran çok daha fazla olduğunu göstermektedir. Regresyon tabanlı yaklaşımlar içerisinde en yaygın olarak kullanılan model ise COCOMO modelidir. İkinci sırada %22 ile İşlev Puanı, üçüncü sırada ise %15 ile uzman görüşüne dayalı çalışmalar yer almaktadır. Bu çalışmanın sonuçlarına göre işlev puanı ile ilgili yapılan çalışmalar 1990'larda zirveye ulaşmış daha sonra azalmaya başlamıştır. Putnam SLIM [3] , Halstead [10] gibi teori tabanlı kestirim yöntemleri 80'li yıllarda çok popüler olmakla birlikte, 1990'lardan sonra popülaritesini kaybetmişlerdir.

2000-2004 yılları arasında yazılım kestirimi için alternatif yöntemlerin kullanıldığı çalışmaların ivme kazandığı görülmektedir. Bu yöntemler arasında Yapay Sinir Ağları, Genetik Programlama, Bulanık Mantık, Lineer Programlama gibi öğrenme tabanlı teknikler sayılabilir. Bu kategori altında yapılan çalışmaların, 2004 yılından sonra da artarak devam ettiği görülmektedir. Örneğin 2007 yılında yapılan bir çalışmada, [11] yapay sinir ağları kullanılarak nesne tabanlı bir yazılımın, sınıf puanları kullanarak çaba kestirimi yapılmaya çalışılmıştır. Regresyon tabanlı modellere göre daha başarılı sonuçlar elde edilmiştir. 2011 yılında genetik programlama yöntemi ile 132 küçük ölçekli proje verisi kullanılarak bir model geliştirilmiş, sonuçlar regresyon tabanlı ve yapay sinir ağları yöntemleriyle kıyaslanmıştır. Sonuçların birbirine yakın olduğu görülmüştür[12]. Yazılım sektörünün gelişimine paralel olarak Türkiye'de de bu alanda çalışmalar yapılmaya başlanmıştır. 2006 yılında yapılan bir çalışmada [13] Türkiye'deki birkaç yazılım firmasının 23 proje verisinden faydalanılarak makine öğrenmesi tekniği ile bir kestirim yapılmaya çalışılmış ve sonuçlar COCOMO II modeli ile karşılaştırılmıştır. 2007 yılında yapılan bir çalışmada, yeni bir yazılım ölçüt kümesi oluşturma, oluşturulan yazılım ölçüt kümesi için veri toplama ve bu veri kümesi ile yapay sinir ağı kullanılarak yeni bir yazılım maliyet tahmini modeli geliştirme çalışmaları yapılmıştır[14]. 2008 yılında yapılan başka bir çalışmada [15], ele alınan projeler için; çalışan kişi sayısı, yazılım büyüklüğü, yazılım kod satır sayısı, bozunma oranı ve yazılım maliyeti gibi veriler toplanarak, bu veriler üzerinden yapay sinir ağları ile bir kestirim yapılmaya çalışılmış ve sonuçlar COCOMO II modeli ile karşılaştırılmıştır.

Yukarıda belirtilen modelleri değerlendirdiğimizde aşağıdaki sonuçlar göze çarpmaktadır:

- Kestirim modellerinin hiç biri tüm projelere uygun değildir. İçinde bulunulan şartlara ve eldeki verilere göre uygun modelin seçilmesi gereklidir.
- Birden fazla yöntemin kombinasyonunu kullanmak daha iyi bir sonuç verebilir. Örneğin uzman görüşü ile yukarıdan aşağı model ve benzeşim bazlı yöntemlerin birleştirilmesi daha iyi bir sonuç verebilir.

- Algoritmik modeller matematiksel modellere dayandığı ve tekrarlanabilir sonuçlar ürettiği için daha objektif bir değerlendirme imkanı sunmaktadır.
- Öğrenme tabanlı çalışmalar özellikle son yıllarda ivme kazanmış ve bu çalışmalarda kayda değer başarılı sonuçlar elde edilmiştir.

Modern yazılım geliştirme dilleri bileşen tabanlı dillerdir. Bu dillerle yeniden kullanılabilir bileşenler geliştirilebildiği için ne İşlev Puanı analiz tekniği, ne de prosedürel diller için kullanılan klasik kestirim yöntemleri bu dillerle geliştirilen uygulamalar için kullanılabilmektedir. Son yıllarda Dördüncü Kuşak Dillerle (4GL) geliştirilen bileşen tabanlı ve form tabanlı uygulamalar için de bazı modeller geliştirilmiştir. Smith [16] bileşen tabanlı yazılım sistemlerinde çaba tahmini yapabilmek için bazı parametreler ortaya koyan bir model geliştirmiştir. Bu model iyi kurgulanmış bir model değildir. Van Koten ve Grey [17] veri tabanı bağlantılı 4GL uygulamalar için bir kestirim modeli sunmuştur. Bu model aşağıdaki kabuller doğrultusunda geliştirilmiştir:

- Bir yazılım sistemi için tamamlanan minimum özellik seti varlık ilişki (Entity Relationship, ER) diyagramı ve fonksiyonel hiyeraşi (FHD) diyagramıdır. FHD sistemin kullanıcı arayüz bileşenlerinin fonksiyonel tanımlarını içerir.
- Sistem veritabanı ER diyagramından otomatik olarak üretilir. Bundan dolayı geliştirme çabası açısından ER diyagramları arasındaki karmaşıklık farkı ihmal edilebilir.
- Sistem üç farklı kullanıcı arayüz bileşeninden oluşur: Formlar, raporlar ve grafikler.

Bu model veritabanı erişimi olan uygulamalar için geliştirilmiştir. Veritabanı erişimi olmayan uygulamalar için uygun değildir. Van Kotel [18] ayrıca veritabanı sistemlerine yönelik Bayes tabanlı istatistiğe dayanan bir çaba kestirim modeli geliştirmiştir. Bu model önceki modelin istatistiksel yaklaşımla geliştirilmiş bir türevidir. Riquelme [19] farklı çaba kestirim yöntemlerini karşılaştırmış ve 4GL uygulamalar için bir model oluşturmuştur. Bu model 4GL uygulamaların metrik kümeleri arasındaki ilişkiyi ve SQL cümleleri kullananlar için bakım zamanlarını analiz etmektedir. Bu model de sadece veritabanı uygulamaları için kullanılabilir. Morgan Peeples [20] bir proje için gereken çabayı hesaplayan bir model geliştirmiştir. Buna göre toplam çaba aşağıdaki formülle belirtilmiştir.

$$LOE_p=a+b_1xForm+b_2xRapor+b_3xTablo+b_4xModül \quad (1)$$

Burada LOE_p P projesi için gerekli olan toplam adam gün eforunu, b katsayıları her nesne için gerekli adam gün sürelerini belirtmektedir. Fakat bu model tüm benzer nesnelerin geliştirilmesi için gerekli olan çabayı eşit kabul ettiğinden gerçekçi bir yaklaşım sunmamaktadır. Gerçek dünyada her bir nesnenin geliştirilmesi için gerekli olan çaba o nesnenin karmaşıklık derecesi ile doğru orantılıdır.

2 Önerilen Çözüm

Bu çalışmada üzerinde çokça çalışılmış ve geniş kabul görmüş bir model olan COCOMO II modelinin tecrübesinden de yararlanılarak nesne yönelimli diller kullanılarak gerçeklenecek form tabanlı yazılım geliştirme projeleri için yeni bir çaba kestirim modeli önerilmiştir. Geliştirilen bu model veritabanı bağlantısı olan form, rapor ve grafiklerden oluşan uygulamalar için kullanılabildiği gibi, veritabanı bağlantısı olmayan uygulamalar için de kullanılabilir. Bu çalışmada COCOMO modelinin temel alınma nedenleri aşağıda sıralanmıştır:

- COCOMO'da kullanılan projeler çeşitlilik göstermektedir. Farklı alanlarda yapılmış projelerdir.
- COCOMO bu alanda oldukça yaygın olarak kullanılmaktadır ve üzerinde pek çok akademik çalışma yapılmaktadır.
- COCOMO'nun ele aldığı projeler iyi ele alınmış projelerdir. İyi seviyede belgelendirilmiş projelerdir.
- COCOMO modeli iyi seviyede belgelendirilmiş bir modeldir. Model ayrıntılı olarak açıktır. COCOMO modeli hem ticari hem de ticari olmayan kurumlarca desteklenmektedir.
- 1981'den 1995'e uzanan bir soy ağacı mevcuttur. Yazılım sektöründeki yaşanan gelişmelere paralel olarak sürekli güncelleştirilmiştir.

COCOMO II ye göre çaba değeri Formül 2'ye göre hesaplanır. Formülde yer alan A katsayısı sabit bir değerdir. COCOMO II'ye 2000 yılında yapılan kalibrasyonda değerinin 2,94 olması önerilmiştir. Büyüklük, KLOC olarak ifade edilen satır sayısının bine bölünmüş değeridir. E katsayısı ise Formül 3'e göre hesaplanır.

$$\text{Çaba}=A \times \text{Büyüklük}^{E} \times \prod_{1}^{17} \text{Çaba Çarpanı} \quad (2)$$

$$E = B + 0.01 \sum_{1}^{5} \text{Ölçek Faktörü} \quad (3)$$

Önerilen modeli geliştirirken COCOMO II'de bir değişken olarak kullanılan satır sayısını elde edebilmek için nesne bazında bir dönüşüm matrisi kullanılması öngörülmüştür. Her bir nesne türü için 1 den 10'a kadar karmaşıklık derecesi öngörülmüş ve her karmaşıklık derecesine karşılık bir satır sayısı tespit edilmiştir. Tablo 1'de verilmiş olan bu matrisin oluşumunda uzman görüşüne başvurularak, uzmanların geçmiş tecrübelerine göre elde edilen satır sayıları minimum ve maksimum değerlerin oluşturulmasında kullanılmıştır. Tablodaki ara değerler ise maksimum ve minimum değerler arasındaki farkın eşit dağılımı şeklinde hesaplanmıştır.

Tablo 1. Nesne Satır Sayısı Dönüşüm Tablosu

	Karmaşıklık Dereceleri				
	1	**2**	**…**	**9**	**10**
Form	10	56	…	385	430
Rapor	10	58	…	401	450

Prosedür	50	116	...	583	650
Entegrasyon	10	44	…	285	320
Tablo	5	16	…	93	105

Bir form tabanlı uygulama için büyüklük, form içerisinde kullanılan nesneler ile modüllerin satır sayısına dönüştürülmüş toplamları ile ifade edilebilir. Bu çalışmada Tablo 2'de yer alan COCOMO II.2000 Mimari Sonrası Evre için kalibre edilmiş ölçüt seti ve değerleri kullanılmıştır.

Tablo 2. COCOMO II.2000 Mimari Sonrası Evre Ölçüt Değerleri

Ölçüt	**Çok Düşük**	**Düşük**	**Normal**	**Yüksek**	**Çok Yüksek**	**Aşırı Yüksek**
Ölçek Faktörleri						
PREC	6,2	4,96	3,72	2,48	1,24	0
FLEX	5,07	4,05	3,04	2,03	1,01	0
RESL	7,07	5,65	4,24	2,83	1,41	0
TEAM	5,48	4,38	3,29	2,19	1,1	0
PMAT	7,8	6,24	4,68	3,12	1,56	0
Çaba Çarpanları						
RELY	0,82	0,92	1	1,1	1,26	
DATA		0,9	1	1,14	1,28	
CPLX	0,73	0,87	1	1,17	1,34	1,74
RUSE		0,95	1	1,07	1,15	1,24
DOCU	0,81	0,91	1	1,11	1,23	
ACAP	1,42	1,19	1	0,85	0,71	
PCAP	1,34	1,15	1	0,88	0,76	
PCON	1,29	1,12	1	0,9	0,81	
APEX	1,22	1,1	1	0,88	0,81	
PLEX	1,19	1,09	1	0,91	0,85	
LTEX	1,2	1,09	1	0,91	0,84	
TOOL	1,17	1,09	1	0,9	0,78	
SITE	1,22	1,09	1	0,93	0,86	0,8
SCED	1,43	1,14	1	1	1	

2.1 Yerel Şartlara Göre Model Kalibrasyonu

Yapılan çalışmalar COCOMO modelinin yerel şartlara göre kalibre edilmesi ile daha başarılı sonuçlar elde edildiğini ortaya koymuştur [7]. Bu çalışmada bu husus göz önüne alınarak çaba formülünde (2) yer alan A katsayısının kalibre edilmesi düşünülmüştür. Kalibrasyon için birden fazla teknik vardır. Bu çalışmada doğal logaritma tekniği kullanılmıştır.

2.2 Verilerin Toplanması

Kalibrasyon yapılabilmesi için veriye ihtiyaç vardır. Bu amaçla telekomünikasyon sektöründe faaliyet gösteren bir firmada Java ve C# dilleri kullanılarak geliştirilen 15 adet ayrı proje kullanılarak elde edilen veriler Tablo 3'te takdim edilmiştir. Bu projelerin 13 tanesi dış kaynak, 2 tanesi ise iç kaynaklarla geliştirilmiştir. Bu projeler altı aylık bir zaman dilimi içerisinde tamamlanmıştır. Projeler geliştirilmeye başlanmadan önce, çaba kestirimi yapabilmek adına ihtiyaç duyulan verilerin toplanabilmesi için bir uygulama geliştirilmiştir. Bu uygulamaya, her bir proje için Tablo 2'de belirtilen çaba çarpanları, ölçek faktörleri ve geliştirilecek olan nesne sayıları ile bunların karmaşıklık değerleri girilmiştir. Bu değerlerin hepsi kurum içinde tutarlılık denetiminden geçirilerek kullanılmıştır. Bu değerlere göre tahmini adam gün süreleri hesaplanmıştır. Hesaplanan bu değerler Tablo 3'te dördüncü sütunda (TÇ) gösterilmiştir. Tüm giriş değerleri ve tahmin edilen adam gün değerleri veritabanına kaydedilmiştir. Projeler tamamlandıktan sonra gerçekleşen adam gün değerleri de veri olarak programa girilmiştir. Bu değerler ise Tablo 3'ün üçüncü sütununda (GÇ) gösterilmiştir.

Tablo 3. Kalibrasyon Tablosu

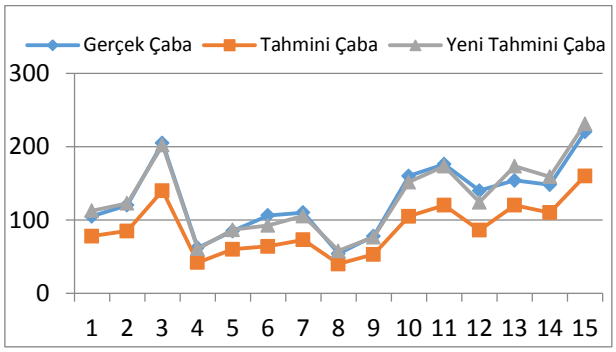
Prj. No	Dil	GÇ	TÇ	Ln (GÇ)	Ln (TÇ/2,94)	Fark	Yeni TÇ
1	Java	105	78	4,65	3,28	1,37	113
2	Java	120	85	4,79	3,36	1,42	123
3	Java	205	140	5,32	3,86	1,46	202
4	Java	62	42	4,13	2,66	1,47	61
5	Java	85	60	4,44	3,01	1,43	87
6	C#	106	64	4,66	3,08	1,58	92
7	Java	110	73	4,70	3,21	1,49	105
8	Java	54	40	3,99	2,61	1,38	58
9	Java	78	53	4,36	2,89	1,46	77
10	C#	160	105	5,07	3,57	1,50	152
11	Java	176	120	5,17	3,71	1,46	173
12	C#	140	86	4,94	3,38	1,56	124
13	Java	154	120	5,04	3,71	1,33	173

14	C#	148	110	4,99	3,62	1,37	158
15	Java	220	160	5,39	3,99	1,40	231

2.3 Kalibrasyon

Kalibrasyon çalışmasında zaman kısıtlamalarından dolayı farklı programla dilleri ile ve farklı alanlarda geliştirilen proje verileri kullanılmıştır. Kalibrasyon için, Tablo 3'teki gerçekleşen çaba (GÇ) değerlerinin logaritmaları alınarak beşinci sütuna yazılmıştır. Daha sonra tahmini çaba (TÇ) değerlerinin A katsayısına bölünmüş olarak logaritmaları alınmış ve altıncı sütuna yazılmıştır. Beşinci ve altıncı sütun arasındaki farklar yedinci sütuna yazılmıştır. Daha sonra bu farkların ortalaması alınarak bir X değeri hesaplanmıştır. Bu X değerinin ters logaritması alınarak ise yeni A katsayısı hesaplanmıştır ($A=e^x$).

Elde edilen yeni A katsayısı kullanılarak 15 adet proje için yeni bir kestirim yapılmış ve elde edilen sonuçlar Tablo 3 de sekizinci sütunda gösterilmiştir. COCOMO II.2000 modelinde mimari sonrası evre için önerilen 2,94 değerine sahip A katsayı yerine, yerel şartlara göre hesaplanmış yeni A katsayısı (A=4,24) kullanılarak tahmin edilen çaba değerlerinin, gerçekleşen çaba değerlerine daha yakın sonuçlar verdiği görülmüştür (Şekil 1).



Şekil 1. Çabaların Karşılaştırılması

Tablo 4. Hata Oranları Tablosu

Prj No	Gerçek Çaba	Tahmini Çaba	Hata Oranı (%)
1	105	112,69	-7,32
2	120	122,80	-2,33
3	205	202,26	1,34
4	62	60,68	2,13
5	85	86,68	-1,98
6	106	92,46	12,77
7	110	105,46	4,12

8	54	57,79	-7,01
9	78	76,57	1,83
10	160	151,69	5,19
11	176	173,36	1,50
12	140	124,24	11,25
13	154	173,36	-12,57
14	148	158,92	-7,38
15	220	231,15	-5,07

Tablo 4'te ise, 15 adet proje için, kalibrasyon yapıldıktan sonra tahmin edilen çaba ile gerçekleşen çabaların karşılaştırılması ve hata oranları gösterilmiştir. Hata oranı (MRE), eşitlik (4)'e göre hesaplanmıştır. Bu tabloda yer alan hata oranlarının mutlak değerlerinin ortalamasının (MMRE) 5,59 olduğu görülmektedir. En yüksek kestirim hatasının ise %12,77 ile 6. projeye ait olduğu görülmektedir.

$$MRE = 100 * (G.\ Çaba - T.\ Çaba) / G.\ Çaba \quad (4)$$

2.4 Kalibrasyon Sonrası Kestirim

Elde edilen sonuçların teyidi ve modelin kendini ispatı için, kalibre edilen A katsayısı ile geliştirilen modelin örnek bir projeye uygulanması düşünülmüştür. Örnek proje Visual C# ile geliştirilmiş bir web uygulamasıdır. Uygulama sözleşme yönetimi için geliştirilmiştir. Yapılan analiz ve tasarım sonucu login ekranı ile birlikte 7 adet giriş ekranı ve 2 adet web servis entegrasyonuna ve 6 adet tabloya ihtiyaç olduğu görülmüştür. Form, entegrasyon, tablo sayıları ve karmaşıklık değerleri Tablo 5' de gösterilmiştir.

Tablo 5. Örnek Proje Verileri

Nesne Tipi	**Karmaşıklık**	**Adet**
Form	2	3
Form	5	3
Form	6	1
Entegrasyon	4	2
Tablo	2	2
Tablo	4	4

Ölçek ölçütlerinden (Scale Factors) TEAM değeri, ekip üyeleri birlikte çalıştığı için "YÜKSEK" olarak seçilmiştir. Proje ekibinin deneyimleri göz önünde bulundurularak personel ölçütlerinden APEX ve PLEX "YÜKSEK" olarak seçilmiştir. Geliştirme aracı olarak Visual Studio 2012 kullanıldığından ve proje

ekibi aynı lokasyonda çalıştığından proje ölçütlerinden TOOL ve SITE ölçütleri "ÇOK YÜKSEK" olarak seçilmiştir. Buna göre Ölçek Faktörlerinin Toplamı PREC + FLEX + RESL + TEAM + PMAT = 17,86999 ve E üssünün değeri E = 0,91 + 0,01 * 17,86999 = 1,0887 olmaktadır. Toplam Satır Sayısı =1,413; 14 adet Çaba Çarpanının değeri$\prod_1^{14} Çaba\ Çarpanı$=0,537176 olmaktadır. Bu değerlere göre (2) eşitliğine göre hesaplanan çaba değeri; Çaba=22 * 4,24 * $1{,}413^{1{,}0887}$ * 0,537176=73,01 adam gün olmaktadır. Gerçekleşen çaba ise 79 adam gündür. Buna göre sapma oranının %8,2 olduğu görülmektedir. Bu değer Tablo 4'te yer alan en yüksek sapma olan %12,77'nin altında bir değer olduğu için modelin bu proje ile yapılan sınamayı geçtiği değerlendirilebilir.

3 Sonuç ve Öneriler

Bu çalışmada çaba kestirim yöntemleri arasında en çok kullanılan COCOMO II.2000 modelinden faydalanılarak, günümüzde yaygın olarak kullanılan form tabanlı uygulamalar için pratik bir çaba kestirim modeli oluşturulmuştur. Ölçüt kümesi içerisinde yer alan özelliklerden TIME, STOR ve PVOL, veri toplama aşamasında kullanılan projeler için geçerli olmadığından bu çalışmada ölçüt kümesinden çıkarılmıştır. 15 adet küçük ve orta ölçekli proje verileri ile gerçekleştirilen bu çalışma sonucunda, modelin ortalama %5,59 oranında sapma ile gerçeğe yakın tahmin yapabildiği görülmüştür. Bu oran yapılan benzer çalışmalara göre oldukça düşük bir orandır. Örneğin Z. Zia, A. Rashid ve K. uz Zaman'ın 2009 yılında geliştirdiği modelle Vb.Net, Visual Basic ve Visual C# ile geliştirilen 19 proje için %11.61 MMRE değeri elde edilebilmiştir. Van Koten modeli bu projeler için uygulandığında ise %17.81 MMRE değeri elde edilebilmiştir [21]. Modelin çekirdek veri kümesini oluşturan 15 projeden dış kaynak kullanılarak gerçeklenen 13 tanesinin farklı firmaların farklı ekipleri tarafından gerçeklenmiş olması, modelin geçerliliğine olumlu katkıda bulunmuştur.

Bu modelin farklı ortamlarda ve farklı organizasyonlarda başarılı sonuçlar sunabilmesi için her ortam için ayrı kalibrasyon yapılmasına ihtiyaç olduğu görülmüştür. Elde edilen sonuçların başarısının seçilen nesne sayılarının ve bunlara ait karmaşıklık derecelerinin doğru bir şekilde seçilmesi ile doğru orantılı olarak arttığı görülmüştür. Daha başarılı sonuçlar elde edilebilmesi için kalibrasyonun geniş bir küme ile ortam bazında, daha homojen verilerle, belirli periyotlarda sürekli olarak yapılması önerilir. Bu çalışmada küçük ve orta ölçekteki projelerden faydalanıldığı için modelin kendini ispatı, eksiklerinin görülmesi ve iyileştirilebilmesi için büyük ölçekli projelerle de çalışma yapılmasına ihtiyaç vardır. Önerilen modeli kendi bünyelerinde kullanmak isteyen ancak büyük ölçekli projeler ile çalışan firmalar, kendi geçmiş projelerinin verileri üzerinden hala önerilen modeli kullanabilirler.

Kaynaklar

1. McCabe, T.J., "A Complexity Measure". *IEEE Trans. on Software Engineering*, vol 2:4, 1976.
2. Gilb, T., "Software Metrics". Winthrop Publishers, 1977.
3. Putman W.L., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem". *IEEE Trans. on Software Engineering*, vol 4:4, 1978.
4. Albrecht, A.J., "Measuring Application Development Productivity". Proc. IBM Applications Development Symposium, Monterey, 1979.
5. Boehm, B.W., "Software Engineering Economics". Prentice Hall, 1981.
6. Londeix, B., "Cost Estimation for Software Development". Addison Wesley, 1987.
7. Boehm, B., Bradford, C., Horowitz, E., Madachy, R., Shelby, R., Westland C., "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0". Annals of Software Engineering Special Volume, Amsterdam, 1995.
8. Boehm, B., Abts, C., Chulani, S., "Software Development Cost Estimation Approaches – A Survey". *Annals of Software Engineering*, 10(1-4), pp 177-205, 2000.
9. Jorgensen, M., Shepperd, M., "A Systematic Review of Software Development Cost Estimation Studies". *IEEE Trans. on Software Engineering*, 33(1), 2007.
10. Halstead, M.H., "Elements of Software Science". Elsevier, 1977.
11. Kanmani, S., Kathiravan, J., Senthil Kumar, S., Shanmugam, M., "Neural Network Based Effort Estimation using Class Points for OO Systems In Computing: Theory and Applications". Int'l. Conf. on Computing: Theory and Applications, Kolkata, 2007.
12. Chavoya, A., Lopez-Martin, C., Meda-Campa, M.E., "Applying Genetic Programming for Estimating Software Development Effort of Short Scale Projects". 8th Int'l. Conf. on Information Technology: New Generations (ITNG), 2011.
13. Baskeles, B., Turhan, B., Bener, A., "Software effort estimation using machine learning methods". 22nd Int'l. Symp. on Computer and Information Sciences, Ankara, 2007.
14. Ayyıldız,M., Kalıpsız, O., Yavuz, S, "Metric-Set and Model Suggestion for Better Software Project Cost Estimation". International Journal of Computer, Information, Systems and Control Engineering, 2008.
15. Sezer A., Okatan, A., "Yazılım Projelerinde Yapay Sinir Ağı Uygulaması ile Maliyet Tahmini". Yüksek Lisans Tezi, Haliç Üniversitesi Fen Bilimleri Enstitüsü, 2008.
16. Smith, R.K., "Effort Estimation in Component Based Software Development: Identifying Parameters". 29th ACM SIGCSE Technical Symp., Atlanta, 1998.
17. Van Koten C., Grey A., "Bayesian Statistical Effort Prediction Models For Data-Centred 4GL Software Development". Dept. of Inf. Science, Univ. of Otago, New Zealand, 2005.
18. Riquelme, J.C., Polo, M., Aguilar Ruiz, J.S., Piattini M., Ferrer Troyano, F. J., Ruiz, F., "A Comparison of Effort Estimation Methods for 4GL Programs: Experiences with Statistics and Data Mining". *Int'l. Jrnl. of Software Engineering and Knowledge Engineering*, vol. 16(1), 2006.
19. Van Koten C., "An effort prediction model for data-centred fourth-generation-language software development". Dept. of Inf. Science, Univ. of Otago, New Zealand, 2004.
20. Peeples, M.J.N., "A software development cost estimation model for higher level language environments". Research paper, 2006. Retrieved from: http://goo.gl/NbBNpm
21. Zia Z., Rashid, A., Zaman, K.uz, "Software Cost Estimation for Component-Based Fourth-Generation-Language Software Applications". *IET Software*, vol 5(1), 2011.