T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM PROJELERİNDE ÇABA KESTİRİMİ İÇİN YENİ BİR YAKLAŞIM

UĞUR KEMAL HAŞLAK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. YUNUS EMRE SELÇUK

İSTANBUL, 2015

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM PROJELERİNDE ÇABA KESTİRİMİ İÇİN YENİ BİR YAKLAŞIM

Uğur Kemal HAŞLAK tarafından hazırlanan tez çalışması 28.01.2015 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Bilimleri Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi _____

Prof. Dr. Oya KALIPSIZ

Yıldız Teknik Üniversitesi _____

Prof. Dr. Nadia ERDOĞAN

İstanbul Teknik Üniversitesi _____

ÖNSÖZ

Bu tez çalışması süresince sonuca gitmede değerli yönlendirmelerini, desteklerini, yardımlarını esirgemeyen değerli hocam Sayın Yrd. Doç. Dr. Yunus Emre SELÇUK'a, değerli katkıları ve yönlendirmeleri için Sayın Prof. Dr. Oya KALIPSIZ ve Sayın Prof. Dr. Nadia ERDOĞAN'a sonsuz teşekkürlerimi sunarım.

Ocak, 2015

Uğur Kemal HAŞLAK

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

4GL	Fourth Generation Language
ER	Entity Relationship
FP	Function Point
LOC	Line of Code
KLOC	Line Of Code/1000
MRE	Magnitude of Relative Error
MMRE	Mean Magnitude of Relative Error
SDLC	Software Development Life Cycle
TB	Toplam Büyüklük (Toplam Satır Sayısı)

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

YAZILIM PROJELERİNDE ÇABA KESTRİMİ İÇİN YENİ BİR YAKLAŞIM

Uğur Kemal HAŞLAK

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Yunus Emre SELÇUK

Yazılım sektörünün gelişimi teknolojinin ve ekonomilerin gelişimine paralel olarak özellikle son yıllarda yüksek bir ivme kazanmıştır. Küçük ve orta ölçekli firmalar bile işlerini yürütebilmek, sektörde rekabet edebilmek için belirli yazılımları kullanmak durumunda kalmaktadır. Büyük ölçekli firmaların ve kamu kuruluşlarının ise ihtiyaçları çok daha farklı olduğu için özel yazılım geliştirme ihtiyaçları çok daha fazladır. Bu ihtiyacı karşılamak için ciddi oranlarda bütçe ayırmaktadırlar. Bu bütçelerin belirlenmesinde yazılım ihtiyaçların belirlenmesinin yanı sıra bu yazılımların maliyetlerinin hesaplanabilmesi de büyük önem arz etmektedir.

Yazılım geliştirme maliyeti her geçen gün artmaktadır. Yazılım maliyet tahmini hem devletler, hem de organizasyonlar için çok önemli bir problemdir. Planlanan zaman ve bütçeyi aşan çok sayıda proje mevcuttur. Bunun temelinde baştan bütçe ve zaman tahminini doğru yapamamaktan kaynaklanan başarısızlıklar yatmaktadır.

Yazılım geliştirme maliyetleri ölçüm ve kestirim yöntemlerinin çeşitli yetersizliklerinden dolayı sık sık kontrol dışına çıkmaktadır. Bu konuda özellikle son 15 yılda çok sayıda çalışma yapılmıştır. Yapılan bu çalışmalar daha çok deneysel olup büyük kısmı uygulanabilir yöntemler ve tatmin edici sonuçlar içermemektedir.

Bu çalışmada bu sorunu hafifletmek amacıyla yeni bir denemede bulunulmuş ve bu alanda yapılan çalışmalardan da faydalanılarak sektörde yaygın olarak kullanılan form tabanlı uygulamalar için kullanılabilecek bir kestirim modeli oluşturulmaya çalışılmıştır. 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 tatmin edici sonuçlar elde edilmiştir.

Anahtar Kelimeler: Yazılım Geliştirme Çaba Kestirimi, Yazılım Geliştirme Maliyet Kestirimi, COCOMO II, Form Tabanlı Uygulamalar

ABSTRACT

A NEW APPROACH FOR SOFTWARE EFFORT ESTIMATOIN

Uğur Kemal HAŞLAK

Department of Computer Engineering

MSc. Thesis

Advisor: Asst. Prof. Dr. Yunus Emre SELÇUK

The development of the software industry has gained a high momentum in parallel with the development of technology and the economy in recent years. Even small and medium-sized firms have to use software to carry out their works and to compete in the industry. Since their needs are very different, large-scale firms and public institutions needs for custom software development much more. To meet this need serious budgets are reserved. Calculating the cost of software is very important for determination of the budget.

Software development costs are constantly increasing. Software cost estimation is a very important problem for organizations and governments. Numerous projects exceed their planned time and budgets. The main cause of this problem is inaccurate estimation of time and budgets.

Software development costs often go out of control due to various deficiencies in measurement and estimation methodologies. In this regard especially in the last 15 years, numerous studies have been made. The majority of these studies are more experimental methods and not practically applicable. Satisfactory results have not been achieved yet.

In order to alleviate the problems stated above, this study proposes a new estimation model for form based applications which are heavily used in industry. The proposed estimation model is based on the most widely used model COCOMO II.2000. In this model, taking into consideration of complexities of objects, estimations can be made by

making use of a conversion table utilizing number of code lines or objects. In this way, a practical and realistic method is constructed. The method is applied to a specific project group and satisfactory results are obtained.

Keywords: Software Development Effort estimation, Software Development Cost estimation, COCOMO II, Form Based Applications

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Gelişen ekonomi ve günümüz rekabet şartları nedeniyle gerek yurtta gerekse dünyada şirketler bankacılıktan otomotiv sanayisine, sağlık bilgi sistemlerinden şirket yönetimine, telekomünikasyon sistemlerinden hava taşımacılığına kadar çok geniş alanlarda en az maliyetle en hızlı şekilde piyasaya ürün çıkarmak ve azami ölçüde müşteri memnuniyeti sağlamak zorunda kalmaktadırlar. Kamu kurumları da değişen hizmet anlayışı ile birlikte vatandaşa en iyi ve en hızlı hizmeti sunabilmek için bilişim teknolojilerinin gücünden daha fazla yararlanma yoluna gitmektedirler. Bu nedenlerle yazılım geliştirme ihtiyacı ve bu alana yapılan yatırımlar her geçen gün artmaktadır.

Yazılım maliyet tahmini hem devletler için hem de şirketler için önemli bir problemdir. Planlanan zaman ve bütçeyi aşan oldukça çok sayıda proje mevcuttur. 1990'larda Standish Group 8000 yazılım projesini incelemiştir [1]. Hem kamu hem de özel projelerden oluşmakta olan bu projelerin %90'ının bütçe ve zaman aşımı nedeniyle başarısız oldukları görülmüştür. Bunların %33'ü daha tamamlanmadan iptal edilmiştir. Yaklaşık üçte biri ilk tahmin edilen ve bütçelenen maliyeti %150 ile %200 arasında ve ortalama olarak %189 aştığı, yaklaşık üçte birinin de %200 ile %300 arasında ve ortalamada %222 aştığı gözlemlenmiştir. Son yapılan çalışmalarda yazılım geliştirme yöntemleri ve proje yönetimi süreçlerindeki gelişmelere paralel olarak bu oranların iyileştiği görülmüştür [2]. Standish Group'un 2013 Chaos Manifesto raporuna göre 2012 yılında yapılan projelerin %39'unun planlandığı gibi başarılı, %43 ünün gecikme ve bütçe aşımı ile tamamlandığı %18 inin ise başarısızlıkla sonuçlandığı görülmektedir.

Gecikme ve bütçe aşımı nedeniyle başarısız kabul edilen projelerin neden bu duruma

düştükleri incelendiğinde, bu projelerin büyük bir kısmı için başlangıçta doğru bir maliyet ve zaman kestirimi yapılamadığı görülmektedir.

1.2 Tezin Amacı

Yazılım geliştirme projelerinin başarısı için maliyet kestirimi çok önemli bir faktördür. Öngörülerin sağlıklı yapılamaması nedeniyle süre aşımları ve yüksek maliyetler ortaya çıkmaktadır. Bir iş için gereken çabayı baştan bilmek çok önemlidir. İşin büyüklüğünü bilmeden işleri yönetmek mümkün değildir. Yönetmek için işin maliyet tahminlerini kullanıp projeleri baştan kabul etmek veya reddetmek çok daha faydalıdır. Çok uzun sürebilecek ve çok kaynak gerektirebilecek bir yazılım geliştirme işine girmemek veya girilme kararı verilse bile baştan gerekli iş gücünü organize etmek, verimi her aşamada artıracaktır. Bu konuda yapılmış çalışmalar olmasına rağmen, hala bu problem tam olarak çözümlenebilmiş değildir ve üzerinde daha fazla çalışma yapılması gerekmektedir.

Bu çalışmanın en önemli amacı mevcut yöntemlerden de faydalanarak günümüz yazılım ihtiyaçlarına ve teknolojilerine uygulanabilecek ve daha isabetli maliyet tahmini yapabilecek bir model geliştirmektir. Yazılım mühendisliğinde başarılı proje; kapsam, zaman ve maliyet hedeflerini yakalamış olan projedir. Bu çalışma maliyet tutturma ve dolayısıyla zaman tahmini yapıp buradaki başarımı artırma konusunda bir fayda sağlayacaktır.

1.3 Hipotez

Bu çalışmada maliyet kestiriminde en sık kullanılan metotlardan biri olan COCOMO II.2000 yöntemi kullanılmıştır. Bu yönteme göre geliştirilen uygulamanın satır sayısını tahmin ederek bir hesaplama yapılmaktadır. Bu satır sayısını tasarım aşamasında tahmin etmek ve bu tahmini objektif kriterlere dayandırmak için, özellikle form tabanlı uygulamalar dikkate alınarak bir nesne / satır sayısı dönüşüm tablosu gerekliliği öngörülmüştür. Bu sayede yapılan tahminlerde sübjektif değerlendirmeler yerine daha gerçekçi ve objektif sonuçlar alınacağı öngörülmüştür.

BÖLÜM 2

YAZILIM ÇABA KESTİRİMİNİN GELİŞİMİ

İlk yazılım ölçüm yöntemlerinden biri satır sayısını sayma yöntemidir. Satır sayısı sayma yöntemi uzun süredir tartışılan bir yöntemdir. Bir programcının verimliliği, satır sayısı sayarak ölçülmeye çalışılmıştır. 70'lerin ortasında geliştirilen çevrimsellik karmaşıklığı (Cyclomatic Complexity) [3] 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 yayınladı [4]. 1978 yılında, SDLC sürecine göre değişkenlik gösteren çaba ve maliyet ihtiyaçlarını dikkate alan Putnam modeli [5] ortaya konmuştur. 1979'da Albrecht, İşlev Puanı'nı (Function Point) [6]; 1981 yılında Boehm, COCOMO modelini tanıtmıştır [7]. Londeix 1987 yılında yaptığı çalışmada makro ve mikro kestirim tekniklerinden bahsetmiştir.[8]. 1995 yılında COCOMO II tanıtılmıştır [9].

Yazılım maliyet kestirim çalışmalarında öncelikle yapılması gereken yazılım geliştirme çabasının kestirimini sağlamaktır. Bunu sağlamak için yazılımın büyüklüğü, sistemin karmaşıklığı ve uygulama alanını göz önünde bulundurmak gerekir. Yazılımın üretilme maliyetini oluşturan tek etken olarak büyüklüğünün düşünülmesi genel bir yanlıştır. Bu yaklaşıma göre tek veri ile maliyet tahmin edilmeye çalışılmakta ve dolayısıyla bütçe aşmaları yaşanmaktadır [10].

$$Ç = f(s) \times g(A) \qquad (2.1)$$

Eşitlik (2.1) ile ifade edilen bu düşünme tarzında Ç:Tahmin edilen Çaba, f(s):Ürünün büyüklüğü, g(A):Ayarlama fonksiyonu olarak simgelenmektedir. Tahmin edilen çaba Adam*Ay olarak ifade edilmektedir. Ancak buradaki başarım oldukça düşüktür. Yazılım

maliyet kestiriminde uzun süre kullanılan bu düşünme tarzında çoğunlukla (2.2)'deki eşitlik kullanılmıştır.

$$\text{Çaba} = A \times (\text{KLOC})^{B} \qquad (2.2)$$

Eşitlik 2.2 ile kaynak kodun satır sayısının kuvvetinin bir düzeltme çarpanı ile beraber gereken çabayı verdiği düşünülmüştür. Belli sayıda proje verileri kullanılarak, matematiksel olarak A ve B sabitleri bulunabilmektedir. Hesapların temelindeki önemli bir nokta, gereken çabanın ürünün büyüklüğünün bir kuvveti şeklinde hızla artmasıdır. Bunun sonucu olarak rastgele yaklaşımla yazılmış bir programın yazılması başta oldukça kolay olmasına rağmen ilerleme oldukça süreç zorlaşmaktadır. Birkaç bin satıra yakın büyüklükteki bir programı bir hafta süresinde yazmak mümkündür. Daha sonra program büyüdükçe hız yavaşlar. Bu program birkaç on bin satıra ulaştığında bir satır ilave etmenin bedeli bir kaç günlük belki de bir kaç aylık çabadır. Dolayısıyla eklemenin oluşturacağı yan etkileri takip etmek zorlaşmıştır. COCOMO modeli ile birlikte büyüklük dışında bazı ölçek faktörleri ve çaba çarpanları da bu hesaplamaya dahil edilmiştir. Bu sayede değişen şartlara göre daha esnek ve gerçekçi kestirimler yapma imkanı doğmuştur.

Temel olarak iki sebepten dolayı yazılım maliyet tahmini yapmak oldukça zordur. Birinci neden yazılımın soyut, elle dokunulamayan, fiziksel olarak alışılmış ürün tanımı dışında olmasıdır. İkinci neden ise yazılım geliştirme işinin fiziksel bir işten ziyade entelektüel bir iş olmasıdır.

2.1 Yazılım Çaba Kestirim Yöntemleri

Yazılım geliştirme çabasını tahmin etmeye yönelik geliştirilen modellerin geçmişi 70'li yıllara kadar dayanır. 2000 yılında yapılan bir araştırmada mevcut kestirim yöntemleri altı başlık altında sınıflandırılmıştır [11]:

Model Tabanlı Teknikler: Matematiksel eşitlikler kullanan, tarihsel veriye ve bir teoriye dayanan teknikler. (Putnam SLIM, Function Point, Estimacs, COCOMO, Checkpoint, SEER)

Uzman Tabanlı Teknikler: Bir alanda bilgi ve tecrübeye sahip uzmanların görüşlerini elde etmeye dayanan teknikler.(Delphi, Rule)

Öğrenme Tabanlı Teknikler: Önceki tecrübelerden ve verilerden yararlanarak kendi kendine öğrenen otomatik sistemler.(Neural, Genetic, Case-based)

Dinamik Tabanlı Teknikler: Çaba faktörlerinin yazılım geliştirme süreci içerisinde değişebildiği öngörüsüne dayanan ve buna uyum sağlayan teknikler.(Abdel, Hamid, Madnick)

Regresyon Tabanlı Teknikler: Model tabanlı tekniklerle birlikte çalışan ve istatistiksel regresyon yaklaşımlarına dayanan teknikler.(OLS, Robust)

Karma Teknikler: Yukarıda belirtilen tekniklerin birkaçını birleştiren teknikler.(Bayesian, COCOMO II)

2002 yılında yapılan bir çalışmada temel olarak yazılım maliyet kestirim yöntemleri iki ana grup altında incelenmiştir[12]: "Algoritmik Modeller", "Algoritmik Olmayan Modeller". Son 10 yılda yapılan çalışmalarla bu sınıflandırmaya üçüncü bir kategori olarak "Öğrenme Tabanlı Modeller" de eklenebilir.

2.1.1 Algoritmik Olmayan Modeller:

Uzman Görüşü: En yaygın olarak kullanılan tekniklerden biridir. Bu modelde tahmin yapabilmek için birkaç uzmanın görüşüne başvurulur. Uzmanlar arasında fikir birliğine varılabilmesi için Delphi veya PERT tekniklerine başvurulur [13]. Kestirim süreci aşağıdaki şekilde uygulanır:

1. Koordinatör her uzmana bir kestirim formu ve kestirim yapacakları projenin bilgilerini iletir.
2. Uzmanlar formu isim yazmadan doldururlar.
3. Koordinatör uzmanları bir toplantı ile bir araya getirerek kestirim yöntemlerini ve kriterlerini tartışmalarını sağlar.
4. Koordinatör uzmanlara tekrar tahmin yapmak üzere form dağıtır.
5. Uzmanlar tekrar formları doldurur ve mantıklı bir sonuç elde edilinceye kadar süreç devam ettirilir.

Benzeşim Yapma: Bu teknik de en yaygın olarak kullanılan tekniklerden biridir[14]. Bu modelde yeni projenin daha önce yapılan benzer projelerle kıyaslanması ve o projelerde

harcanan çaba dikkate alınarak mantık yürütme sonucu bir tahmin yapmaya dayanır. Bu teknikte de uzmanların görüşüne başvurulur. Aşağıdaki adımlar uygulanır:

1. Yeni projenin karakteristiği çıkarılır.
2. Tamamlan projelerden yeni projeye karakteristik olarak en uygun projeler seçilir.
3. Bu projelere benzeşim yapılarak yeni projenin ne kadarlık bir çaba ile hayata geçirilebileceği tahmin edilmeye çalışılır.

Yukarıdan Aşağıya Kestirim Yöntemi: Bu yöntem Makro Model olarak da bilinir. Projenin başlarında bir kestirim yapma ihtiyacı varsa kullanılabilir [12]. Proje gereksinimleri üst seviyede ortaya çıkmıştır ve henüz detaylı analiz çalışması yapılmamıştır. Putnam modeli [5] de bu yaklaşımı kullanır. Bu yöntemle hızlı kestirim yapmak mümkündür fakat kestirimin doğruluk oranı düşük olabilmektedir.

Aşağıdan Yukarıya Kestirim Yöntemi: Bu yöntem, bütünü meydana getiren bileşenleri ayrı ayrı ele alarak bir kestirim yapmaya çalışır [12]. Bileşenlerin tahmini süreleri toplanarak tüm uygulamayı geliştirmek için gereken toplam çaba hesaplanmaya çalışılır. Bu yöntem detaylı analiz yapıldıktan sonra kullanılabilir. Bu yöntemle daha doğru sonuçlar elde edilebilir fakat detaylı analiz gerektirdiği için hızlı bir kestirim yöntemi değildir. COCOMO'nun detaylı modeli de bu yaklaşımı kullanmaktadır.

Parkinson Yöntemi: Parkinson prensibine göre bir işi yapmak için gereken çabayı elimizde bulunan imkanlara ve kısıtlara göre tahmin edebiliriz [12]. Örneğin bir yazılımın 12 ayda bitmesi gerekiyorsa ve elimizde 5 kaynağımız varsa bu yazılımın hayata geçirilmesi için gereken tahmini çaba 60 adam aydır. Aslında bu bir tahmin değil bir tespitdir. Bu nedenle de hedefin tutması pek olası değildir.

Kazanmak için Fiyat Yöntemi: Bu yönteme göre iş yapabilmek için en iyi fiyatı vermek esastır [12]. Müşterinin bütçesi belli bir miktar ile sınırlı ise bu bütçeye uymak için bir tahmin yapılır. İşin büyüklüğü ve işlevlere göre bir kestirim yapılmadığı için bu yöntemin de başarılı sonuçlar vermesi pek mümkün değildir.

2.1.2 Algoritmik Modeller

Algoritmik modeller matematiksel modellere dayanan, önemli maliyet faktörlerini değişken olarak kullanan fonksiyonlarla çaba tahmini yapmaya çalışan modellerdir.

Çaba= $f(x_1, x_2, x_3,\ldots\ldots x_n)$ formunda fonksiyonlardan oluşurlar. Birbirlerinden ayrılma nedenleri, fonksiyonların oluşma formu ve kullanılan çaba faktörlerinin farklılıklarıdır. En bilinen modeller Putnam ve COCOMO modelleridir. Algoritmik modeller sınıflandırıldığında Çizelge 2.1'de gösterildiği gibi bir sınıflama ortaya çıkmaktadır.

Çizelge 2.1 Algoritmik modellerin sınıflandırılması [12]

	Lineer	**Çarpanlı**	**Üssel**	**Ayrık**	**Diğer**
Gözlemsel	Nelson	Walston-Felix	COCOMO	Aron, Boeing, Wolverton	Prise-S
Analitik			Putnam		Softcost

2.1.2.1 Putnam Metodu

Putnam metodunda, sistem geliştirme sürecini zamana bağlı olarak gösteren çaba ve maliyet eğrileri bulunmaktadır. Gerçekçi bir projede hesaplanan adam ay değeri, süreç boyunca sabit kalmaz. Dolayısıyla bazı ayların personel gereksinimi, diğerlerine göre farklı olacaktır. Bu çaba-zaman eğrisi gözlenerek personel istihdamı ayarlanabilir. Aynı kurum içerisinde farklı projeler arasında eleman kaydırmaları da Putnam eğrilerinin bir sonucu olarak mümkün olur [5]:

$$\text{Çaba} = ((\text{Satır Sayısı}) \times (\text{ÖY}^{0.333} / \text{VF}))^3 \times (1 / \text{PS}^4) \quad (2.3)$$

(2.3) eşitliğinde PS proje süresi, ÖY özel yetenekler katsayısıdır. 5.000 ile 15.000 satırdan oluşan küçük projeler için ÖY değeri 0.16 iken, 70.000 satırlık veya daha büyük projeler için 0.39 dur. VF ise verimliliğe bağlı olarak değişir: Gerçek zamanlı sistemler için 2.000 olan değer, sistem programları ve iletişim programları için 10.000 ve iş bilgi sistemleri için 28.000 dir. Verimliliğe etki eden bazı faktörler aşağıda verilmiştir.

- Geliştirilen uygulamanın karmaşıklığı
- Yazılım geliştirme ortamı
- Takımın yetenekleri ve deneyimi
- Yazılım mühendisliği uygulanmasının niceliği
- Kullanılan programlama dilinin soyutlama düzeyi
- Süreç olgunluğu ve yönetim uygulamaları

2.1.2.2 COCOMO Modeli

1981 yılında, Dr. Barry Boehm "Software Engineering Economics" adlı kitabında COCOMO'yu (Constructive Cost Model) tanıtmıştır[7]. Oldukça ilgi görmüş bir maliyet kestirim modelidir. Gereken çabanın, program büyüklüğünün bir üstel değerine bağlı olması prensibi ve endüstriden toplanan bilgiler ışığı altında geliştirilmiş bir kestirim metodu olan COCOMO, deneysel ve gözlemsel bir modeldir. COCOMO 81, birçok yazılım projesinden veri toplanarak, ardından gözlemlerle uyuşacak en iyi formülü keşfetmek için bu verilerin analiz edilmesi ile çıkarılmış bir modeldir. 1987 yılında, Yazılım Mühendisliği Enstitüsünde yapılan 3. COCOMO kullanıcı grup toplantısında Ada COCOMO ve artımsal COCOMO tanıtıldı. 1988, 1989 yollarında Ada COCOMO'da geliştirmeler tanımlandı.

COCOMO Modeli ile bir kestirim yapabilmek için öncelikle LOC olarak ifade edilen satır sayısını tahmin edebilmek lazımdır. Bu tahmini yapabilmek oldukça zordur. Programcıların tecrübesine, kullanılacak olan programlama diline; modüler, nesne tabanlı, prosedürel programlama gibi tekniklerin kullanılma durumuna; servis tabanlı mimari, çok katmanlı mimari, istemci sunucu mimarisi gibi mimari yaklaşımlardan hangisinin benimsendiğine göre, geliştirilecek olan kodun satır sayısı da değişkenlik gösterecektir. Ayrıca modern yazılım geliştirme araçları otomatik olarak birçok kodu kendi üretmektedirler. Bu noktada aşağıda belirtilen teknikler ortaya çıkmaktadır.

Uzman Görüşüne Göre Satır Sayısı Kestirimi: Bu yöntemde benzer büyüklükte ve alanda proje geliştirmiş olan uzmanlardan geliştirilecek olan programın satır sayısını tahmin etmeleri istenir. Kestirim yapılırken "Yukarıdan Aşağıya" ya da "Aşağıdan Yukarıya" teknikleri kullanılabilir. Uzmanların ortak görüşünü elde edebilmek için Delphi ya da PERT tekniği kullanılabilir. Tamamlanmış proje verilerinden de faydalanılabilir.

Nesne - Satır Sayısı Dönüşümü ile Satır Sayısı Kestirimi: Bu yöntem özellikle kullanıcı ara yüzlerinin bulunduğu form tabanlı uygulamalar için kullanılabilir bir yöntemdir. Form, rapor, tablo, web servisi gibi somut nesne ve bileşenlerin karmaşıklık derecesine göre satır sayıları tespit edilmeye çalışılır. Bu tespitler yapılırken minimum ve maksimum karmaşıklık dereceleri için uzman görüşüne ve geçmiş proje verilerine bakılır. Ara değerler eşit dağılım şeklinde hesaplanır. Bu çalışmada da satır sayısı kestirimi için bu yöntem kullanılmıştır.

İşlev Puanı Yöntemi ile Satır Sayısı Kestirimi: Eğer proje ile ilgili girdi çıktı gibi özellikler tahmin edilebiliyorsa bunlar kullanılarak geliştirilecek sisteme ait satır sayısı tahmin edilebilir. Bunun için Çizelge 2.2'de gösterilen dönüşüm tabloları kullanılır.

Çizelge 2.2 İşlev Puanından Satır Sayısına Dönüşüm Tablosu [15]

Programlama Dili	Ortalama	Medyan	En Az	En Fazla
Access	35	38	15	47
Ada	154		104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Cobol	77	77	14	400
Java	63	53	77	
JavaScript	58	63	42	75
JSP	59			
Oracle	30	35	4	217
Perl	60			
PL/I	78	67	22	263
PowerBuilder	32	31	11	105
SQL	40	37	7	110
VisualBasic	47	42	16	158

İşlevsellik doğrudan ölçülemeyeceğinden, bir yazılım projesinde işlevselliğe etkisi olan birçok etken bir arada incelenerek ürüne olan etkileri ağırlıklandırılabilir. Sonuç olarak bir sayı ortaya çıkar ve bu rakam değişik projeleri göreceli olarak değerlendirmekte yararlı olabilir. İşlev puanını satır sayısının yerine konarak diğer dolaylı ölçümlere ulaşılabilinir. İki ayrı yaklaşımı birleştirici dönüştürme tabloları bulunmaktadır. Satır sayılı veya işlev puanlı ölçmelerde birinden diğerine geçmek, kabaca değerlerle mümkündür. Çizelge 2.2 bu çevrimi değişik programlama dilleri açısından vermektedir. Çizelge 2.2 sayesinde elimizde bulunan bir işlev puan için ortalama kaç satır kod gerektiği bulunabilinmektedir. Çizelge 2.2'de bulunan dönüştürme nesne tabanlı değil, prosedürel yaklaşıma göre oluşturulmuştur. Nesne tabanlı yaklaşımda gerekli satır sayısı ciddi oranda azalmaktadır. Ancak satır sayısının azalması gereken iş gücünün

doğrudan azalması veya artması anlamı taşımamaktadır.

İşlev puanı hesaplamak için karmaşıklığa etki eden faktörlerin ağırlıklandırılmaları ve sonuçta karmaşıklığa olan etkileri denenerek bu ağırlıkların ayarlanması şeklinde yöntemler uygulanmıştır. Bulunan sonuçları, verilen formüllere proje ile ilgili doğrudan ölçebileceğimiz büyüklükleri girerek kullanabiliriz. Bunun için Çizelge 2.3'deki gibi bir hesaplama çizelgesi kullanılabilir.

Çizelge 2.3 İşlev Puan hesaplama çizelgesi

Ölçme parametresi	**Adet**	**Basit**	**Orta**	**Karmaşık**	**Sonuç**
Kullanıcı girdi sayısı	…	3	4	6	…
Kullanıcı çıktı sayısı	…	4	5	7	…
Sorgu sayısı	…	3	4	6	…
Kütük (dosya) sayısı	…	7	10	15	…
Dışsal arayüz sayısı	…	5	7	10	…
Toplam Sayı	…	…	…	…	…

İşlev puanı hesaplama için iki işlem yapmak gerekir. Önce "Toplam Sayı" denilen nesnel ölçümlere dayanan değeri bulmak gerekmektedir. Dışsal arayüz sayısı, kütükler ve bunun gibi her türlü makine tarafından anlaşılan bilgi aktarım noktaları sayısıdır. Kullanıcı girdileri, kullanım sırasındaki veri girişleridir. Toplam sayı, adet sütununda belirtilen sayılar ile ilgili karmaşıklık katsayısının çarpılması ve sonuçların toplanması sonucu elde edilir. Toplam sayı bu şekilde bulunduktan sonra işlev puan (2.4)'deki eşitlik ile hesaplanır.

$$\text{İ}P \;=\; (Toplam\ Sayı)\ x\ (0.65 \;+\; (0.01\ x\ K_i)) \qquad (2.4)$$

Eşitlik (2.4)'te 14 adet K_i "Karmaşıklık Ayarlama Faktörü" mevcuttur ve aşağıdaki belirtilen soruların cevaplarından meydana gelir. Bu sorulara verilecek cevaplar 0 ile 1 arasında değer alır. Böylece ortaya çıkacak 14 değer toplanır ve 0.01 ile çarpılarak formüle konur. Karmaşıklık Ayarlama Faktörleri (K_i) şu şekilde listelenir [16] :

- Güvenilir yedekleme ve kurtarma işlemi gerekli mi?
- Veri iletişimi gerekiyor mu?
- Dağıtık işlem ve süreçler var mı?

- Çabukluk önemli mi?
- Sistem mevcut ve fazla yüklü bir ortamda mı çalışacak?
- Çevrimiçi (on-line) veri girişi gerekecek mi?
- Çevrimiçi (on-line) giriş, fazla ekranlı veya fazla işlemli mi?
- Ana kütükler çevrimiçi (on-line) olarak mı güncellenecek?
- Girdi, çıktı, sorgulama ve kütükler karmaşık mı?
- İç süreç (internal process) karmaşık mı?
- Program yeniden kullanılabilir olarak mı tasarlanıyor?
- Dönüştürme (conversion) ve kurma (installation), tasarımın içinde yer alıyor mu?
- Değişik kuruluşlarda çoklu kurmalar tasarlanıyor mu?
- Kullanıcının kolaylığı ve uyarlamasına göre tasarlanıyor mu?

COCOMO 81

COCOMO modelinin ilk versiyonudur. Maliyet tahmini analizinin ayrıntılarını yansıtan üç seviyeli bir modeldir [7].

- Birinci seviye: Başlangıç ve kaba bir tahmin sağlanmakta
- İkinci seviye: Proje ve süreç çarpanlarını kullanarak bunu değiştirir
- Ayrıntılı seviye: Projenin farklı aşamaları için tahminler üretir

Yapılacak hesapların kapsamları açısından üç değişik modelden oluşur. Basit model, orta ve detaylı modeller. Ayrıca bu modellerde kullanılacak problemler organik, yarı ayrık ve gömülü sınıflar altında toplanmıştır:

- Organik problemler için küçük boyuttaki programcı takımları, bildikleri ortamlarda iyi anlaşılmış uygulamaları geliştirirler. İletişim problemleri azdır ve elemanlar hızlıca işlerini halledebilecek durumdadırlar.

- Yarı ayrık problemlerde ise ekipte tecrübeli ve tecrübesiz elemanlar bulunabilir. İlgili sistemler konusunda deneyimleri sınırlı olabilir ve geliştirilen sistemin her kısmını bilmeyebilirler.
- Gömülü problemler: Geliştirilecek yazılım, sistemin donanım, kurallar, işletim süreçleri veya yazılım gibi diğer bileşenleri ile çok kuvvetli bağlantılar oluşturur. Gereksinim değişiklikleri ile problemleri halletmek oldukça zordur. İhtiyaç belirtiminin geçerlilik incelenmesi çok maliyetlidir. Yazılımın geliştirilmesinde görev alanların tüm kısımlara hâkim olma olasılığı iyice azalmıştır.

Çizelge 2.4 Basit COCOMO Modelleri

Problem	**Çaba**	**Süre**
Organik	Çaba = 2.4 $(KLOC)^{1.05}$	Süre = 2.5 $(Çaba)^{0.38}$
Yarı ayrık	Çaba = 3 $(KLOC)^{1.12}$	Süre = 2.5 $(Çaba)^{0.35}$
Gömülü	Çaba = 3.6 $(KLOC)^{1.20}$	Süre = 2.5 $(Çaba)^{0.32}$

COCOMO bu model ve problem sınıfı saptamalarından sonra ortaya çıkan formüllerle tahmin hesaplama yolunu önerir. Çizelge 2.4'de problem sınıflarına göre basit model için formüller gösterilmektedir. Orta detaydaki model ise sistemin güvenilirlik, veri tabanı büyüklüğü, işletme ve kayıt sınırlandırmaları, personel özellikleri ve kullanılan yazılım araçları gibi diğer özelliklerinin hesaba katılması amaçlanmıştır.

Çizelge 2.5 Orta Detayda COCOMO Çaba Formülleri

Problem	**Çaba**
Organik	Çaba = 3.2 $(KLOC)^{1.05}$ x ÇAK
Yarı ayrık	Çaba = 3.0 $(KLOC)^{1.12}$ x ÇAK
Gömülü	Çaba = 2.8 $(KLOC)^{1.20}$ x ÇAK

Çizelge 2.5 ve Çizelge 2.6'da orta detaydaki COCOMO çaba formülleri ve ölçütleri gösterilmiştir. Belirli bir takım özelliğin, proje açısından etkileri ayrı ayrı ağırlandırılarak katsayılar ortaya çıkarılır. Çizelge 2.7'de yazılım geliştirme çaba çarpanları gösterilmiştir. Bu faktörler, ilgili özellik için düşük (<1), nominal (1) veya yüksek (>1) olarak saptanırlar.

Katsayılar birbiri ile çarpıldığında “Çaba Ayarlama Katsayısı” (ÇAK) (Effort Adjustment Factor) bulunur. Bu katsayı 0.9 ile 1.4 arasında bir değer alır ve Çizelge 2.5’de gösterilen Orta COCOMO modeli formülleri kullanılarak çaba hesaplaması ile sonuçlandırılır. Süre hesaplaması ise Basit COCOMO modelinde olduğu gibi yapılır.

Çizelge 2.6 Orta Detayda COCOMO Ölçütleri

Ürün	**Donanım**	**İnsan**	**Proje**
Güvenilirlik	Hız	Analist yeteneği	Yazılım aracı Kullanımı
Veri tabanı büyüklüğü	Bellek yeterliliği	Uygulama Deneyimi	Zamanlandırma
Ürün Karmaşıklığı	Sanal Makine Değişkenliği	Geliştirme Ortamı Deneyimi	Yeni Programlama Teknikleri
	Kullanılabilme Süresi	Yazılım Geliştirici Yeteneği	
		Programlama Dili Deneyimi	

Detaylı COCOMO modeli ise projenin evrelerine bağlı olarak süreç içinde değişiklikleri hesaba katarak arada bir kestirim hesaplamasını önerir. Bu modelde zamana bağlılık temel değişikliktir. Projenin farklı evrelerinde çaba yoğunluğu ve yapılacak işin karmaşıklığı değişecektir. Benzer bir anlayış, yazılım eğrisi denilen bir sonucu yansıtan Putnam modelinde de kendisini gösterir. Evrelere bağlılığın bir sonucu olarak Raleigh adam-ay eğrileri ortaya çıkmıştır. Proje başlangıcındaki az iş gücü ile gereksinimlerin ilk çabalarını düşük düzeylerle temsil edildiğini bu eğrilerde görebiliriz. Hızla tırmanan eğri, geliştirmenin analiz, tasarım ve uygulama gibi evrelerinde yükseklerdedir. Daha sonra geliştirme sonrası faaliyet azalır. İleride bakım ve onarım yine bazı geçici yükselmeler yapabilir.

Çizelge 2.7 Yazılım Geliştirme Çaba Çarpanları

Ölçüt	Açıklama	Değerlendirme					
		Çok Az	Az	Normal	Yüksek	Çok Yüksek	Aşırı Yüksek
Ürün							
RELY	Yazılımın Güvenilirliği	0.75	0.88	1.00	1.15	1.40	-
DATA	Veritabanı Hacmi	-	0.94	1.00	1.08	1.16	-
CPLX	Ürün Karmaşıklığı	0.70	0.85	1.00	1.15	1.30	1.65
Bilgisayar							
TIME	Çalışma zamanı kıstı	-	-	1.00	1.11	1.30	1.66
STOR	Ana depolama kıstı	-	-	1.00	1.06	1.21	1.56
VIRT	Sanal makine uçuculuğu	-	0.87	1.00	1.15	1.30	-
TURN	Bilgisayar değişim (dönme) zamanı	-	0.87	1.00	1.07	1.15	-
Personel							
ACAP	Analistin yetenekleri	1.46	1.19	1.00	0.86	0.71	-
AEXP	Uygulama Tecrübesi	1.29	1.13	1.00	0.91	0.82	-
PCAP	Programcı kapasitesi	1.42	1.17	1.00	0.86	0.70	-
VEXP	Sanal Makine Tecrübesi	1.21	1.10	1.00	0.90	-	-
LEXP	Dil Tecrübesi	1.14	1.07	1.00	0.95	-	-
Proje							
MODP	Modern programlama deneyimi	1.24	1.10	1.00	0.91	0.82	-
TOOL	Yazılım Araçları	1.24	1.10	1.00	0.91	0.83	-
SCED	Geliştirme Zaman Planı	1.23	1.08	1.00	1.04	1.10	-

COCOMO II

1995 yılında COCOMO II'yi tanıtan makaleler yayınlanmıştır [9]. 1997 ve 1998 yıllarında COCOMO II için ilk kalibrasyonlar yapıldı. 2000 yılında son kalibrasyon yapıldı. COCOMO II.1998 ismi COCOMO II.1999 olarak sonra COCOMO II.2000 olarak değiştirildi. Aslında üçü de tamamen aynıdır.

COCOMO 81, yazılımın şelale sürecine göre geliştirildiği düşünülerek geliştirilmiş bir modeldir. Bu başlangıç versiyonunun önerilmesinden sonra yazılım geliştirmede büyük değişikler meydana geldi. Yazılım, yeniden kullanılabilir bileşenlerin birleştirilmesi ve

bunların bir betik dil aracılığıyla geliştirilmesi yoluyla yaratılabilir. Prototip oluşturarak ve artırımsal geliştirme çoğunlukla kullanılan bir süreç modelleridir. Birçok durumda var olan alt sistemler, kütüphaneler kullanılır. Yeni yazılım oluşturmak için var olan yazılım üzerine yeniden mühendislik yapılabilmektedir.

Bu değişiklikleri hesaba katmak için COCOMO II prototip oluşturarak, bileşenleri birleştirerek geliştirme, dördüncü nesil programlama dillerini kullanma gibi farklı yaklaşımlar kabul etmiştir. Model seviyeleri, artan bir şekilde karmaşık ve detaylı tahminleri yansıtmamaktadır. Seviyeler, yazılım sürecindeki aktivitelerle ilişkilendirildiğinden, başlangıç tahminleri, sistem mimarisi tamamlandıktan sonra gerçekleştirilen daha detaylı tahminle birlikte süreçte erken yapılırlar. COCOMO II'de tanımlanan seviyeler:

- **Erken prototip oluşturma seviyesi:** Büyüklük tahminleri işlev puanlarını temel alır ve gerekli çabayı kestirmek için basit büyüklük/üretkenlik formülü kullanılır.
- **Erken tasarım seviyesi:** Bu seviye, sistem ihtiyaçlarını bazı başlangıç tasarımıyla tamamlamaya karşılık gelir. Tahminler, kaynak kodun satır sayılarına dönüştürülecek işlev puanını temel alır. Formülde standart formüle ek olarak ona ilişkilendirilmiş bir çarpan kümesi mevcuttur.
- **Mimari sonrası seviyesi:** Sistem mimarisi tamamlandığında, yazılım büyüklüğü hakkında mantıklı, doğru tahmin yapılabilir. Bu seviyedeki tahmin, personelin yeteneklerini, ürün ve proje özelliklerini yansıtan daha kapsamlı çarpan kümesi kullanır.

Erken Prototip Oluşturma Evresi:

Erken prototip oluşturma evresi, prototip oluşturularak gerçekleştirilen projeler ve var olan bileşenlerin birleştirilmesiyle oluşturulan projeler için gerekli çabayı kestirmek amacıyla COCOMO'nun içinde mevcuttur [10]. Tahmin edilen üretkenlik için standart sayıya bölünen ağırlıklı işlev puanları temel alınır. Programcı üretkenliği; geliştiricinin deneyimine ve yeteneğine ve geliştirmeyi desteklemek için kullanılan bilgisayar destekli yazılım geliştirme araçlarının yetenekleri ile ilişkilidir. Çizelge 2.8'de üretkenliğin farklı seviyeleri gösterilmektedir [9].

Çizelge 2.8 COCOMO modelinde üretkenlik [9]

Yazılım Geliştiricinin Deneyimi ve Yeteneği	Bilgisayar Destekli Yazılım Mühendisliği Aracının Olgunluğu ve Yeteneği	Üretkenlik
Çok Az	Çok Az	4
Az	Az	7
Normal	Normal	13
Yüksek	Yüksek	25
Çok Yüksek	Çok Yüksek	50

Bu seviyede, yeniden kullanım oldukça yaygındır. Bu yüzden plan tahmininde kullanılan işlev puanları beklenen yeniden kullanım yüzdesini hesaba katarak ayarlanmalıdır.

Çaba=(İşlev puan sayısı x (1 - %Yeniden Kullanım/100)) / üretkenlik (2.5)

Erken tasarım Evresi:

Erken tasarım evresinde üretilen tahminler, algoritmik modeller için standart formülü temel alırlar. Erken tasarım seviyesinde çaba (2.6)'daki şekilde hesaplanabilmektedir.

$$\text{Çaba} = A \; x \; \text{Büyüklük}^{E} \; x \; M \qquad (2.6)$$

Boehm, A katsayısının bu seviyede yapılan tahminler için 2,5 olmasını önermiştir. Sistemin büyüklüğü kaynak kodun binler cinsinden satır sayısı olarak ifade edilir. Bu büyüklük tahmini, üreteçlerle yaratılan veya yeniden kullanılan kod yerine el ile gerçekleştirilen kodu ifade eder.

E üssü, projenin büyüklüğü arttıkça, artan gerekli çabayı yansıtır. Bu COCOMO'nun ilk versiyonundaki gibi sabit değildir ama projenin yeniliğine, geliştirme esnekliğine, kullanılan risk çözünürlük süreçlerine, geliştirme takımının bütünlüğüne ve organizasyonun süreç olgunluk seviyesine bağlı olarak 1.1 ile 1.24 arasında değişebilmektedir.

M çarpanı, ürün güvenilirlik ve karmaşıklığı, gerekli olan yeniden kullanım, platform zorluğu, personel yeteneği, personel deneyimi, plan ve destek araçlarını içeren yedi proje ve süreç sürücülerini içeren basitleştirilmiş kümeyi temel alır. Bunlar, 1'in düşük değerlere karşılık geldiği, 6'nın yüksek değerlere karşılık geldiği 6 noktalı ölçekle tahminlenirler. Alternatif olarak, daha ayrıntılı çarpanların kullanıldığı mimari sonrası

seviyesindeki değerler birleştirilerek de hesaplanabilir. Çaba hesaplama (2.7) ve (2.8)'deki gibi yapılabilir.

$$\text{Çaba} = A\ x\ \text{Büyüklük}^{E}\ x\ M\ +\ PMm \qquad (2.7)$$

M = PERS x RCPX x RUSE x PDIF x PREX x FCIL x SCED

(2.7)'de ki PMm, kodun otomatik olarak yaratılması durumunda kullanılan faktördür. Bu yüzden, gerekli çaba (PMm) eşitlik (2.8) kullanılarak ayrıca hesaplanır ve el ile geliştirilen kod için hesaplanan çabaya eklenir.

PMm= (ASLOC x (AT/100)) / ATPROD (2.8)

ASLOC, otomatik olarak yaratılmış kaynak kodun satır sayısıdır. ATPROD, bu tipteki kod üretiminin üretkenlik seviyesidir. Üreteçlerle yaratılan kodun sistemin kalan kısmıyla arayüzünü oluşturmak için ek olarak çaba gerekmektedir. Bu, otomatik olarak oluşturulan kodun (AT), toplam sistem koduna yüzdesine bağlıdır. Gerçek üretkenlik, üreteçlerle yaratılan modül sayılarına bağlıdır. Üreteçlerle yaratılan kodun miktarı ne kadar azsa, bunu sistemin içindeki diğer kodlarla uyuşturmada ve tümleştirmede daha fazla çaba gerekmektedir.

Mimari sonrası evresi:

Mimari sonrası evresinde üretilen tahminler, erken tasarım tahminlerinde kullanılan aynı temel formülü temel alırlar. Bununla birlikte, yazılım için büyüklük tahmini kestirim sürecinde bu evre ile daha doğrudur ve başlangıç çaba hesabı yedi özellik yerine 17 özellik kullanılarak arttırılmıştır. Buna göre tahmini çaba eşitlik (2.9) de belirtildiği şekilde hesaplanır.

$$\text{Çaba}\ =\ A\ x\ \text{Büyüklük}^{E}\ x\ \prod_{1}^{17} \text{Çaba Çarpanı} \qquad (2.9)$$

Kaynak koddaki toplam satır sayısının tahmini iki önemli proje faktörünü hesaba katarak ayarlanmıştır:

- **Mümkün olabilecek yeniden kullanımın büyüklüğü:** Kapsamlı yeniden kullanım, geliştirilen kaynak kodun satırlarının sayısının değiştirilmesinin gerekli olduğu anlamına gelir. Bununla birlikte Selby M., yeniden kullanım maliyetlerinin doğrusal olmadığını keşfetti [17]. Bunun nedeni, gerekli olan bileşenleri keşfetmek ve seçmek için harcanan başlangıç çabası ve yeniden kullanılabilir bileşenlerin

değiştirilmeye ihtiyacı varsa, onların ve ara yüzlerinin anlaşılması için gerekli çabadır.

- **İhtiyaçların değişkenliği:** Tahmin, sistem ihtiyaçlarındaki değişiklikleri uzlaştırmak için gerekli yeniden çalışmadan oluşur. Bu tahmin, değiştirilmesi gerekli kaynak kodun satırlarının sayısı cinsinden ifade edilir ve bu sayı başlangıç büyüklük tahminine eklenir.

Yeniden kullanımın etkileri COCOMO II'de büyüklük çabasını ayarlayarak eşitlik (2.10)' a göre hesaba katılmıştır.

ESLOC=ASLOC x (AA + SU + 0.4DM + 0.3CM + 0.3IM) / 100 (2.10)

ESLOC, yeni kodun denk satırlarının sayısıdır. ASLOC, değiştirilmesi gerekli yeniden kullanılabilir kodun satırlarının sayısıdır. DM, değiştirilen tasarımın yüzdesidir. CM, değiştirilen kodun yüzdesidir. IM, yeniden kullanılabilir yazılımı tümleştirmek için gerekli orijinal tümleştirme çabasının yüzdesidir. SU, yazılım anlaşılabilirliğini temel alan bir değerdir. 50 (karmaşık, yapısal olmayan kodlar için) ile 10 (iyi yazılmış, nesneye dayalı kod) arasında değerler alabilir. AA, yazılımın yeniden kullanılabilir olabileceği hakkında verilen karar için başlangıç değer biçme maliyetlerini yansıtan bir değer olup, yapılan test miktarına ve gerekli olan değerlendirme miktarına bağlıdır. Değeri 0'dan 8'e kadar değişebilir.

Üstel terim, COCOMO'nun ilk sürümünde, çaba hesaplama formülünde 3 farklı değere sahip olabilmekteydi. Bunlar farklı seviyelerdeki proje karmaşıklığıyla ilişkilendirilmişti. Projeler karmaşıklaştıkça, artan sistem büyüklüğünün etkileri daha önemli hale gelmektedir. Bununla birlikte, organizasyonel uygulamalar ve prosedürler bu ekonomik olmayan ölçeği kontrol edebilirler ve bu COCOMO II'de kabul edilmişti. Üs, aşağıdaki şekilde gösterildiği gibi beş adet ölçek faktörü göz önüne alınarak hesaplanır.

- Sürecin olgunluğu (PMAT)
- Geliştirme esnekliği (FLEX)
- Benzer proje tecrübesi (PREC)
- Mimari/risk çözünürlüğü (RESL)
- Takım bütünlüğü (TEAM)

Bu faktörler, altı dereceli ölçekte sınıflandırılırlar. Üstel terimi elde etmek için sonuçlanan oranlar toplanır, 100'e bölünür ve sonuca sabit bir katsayı ilave edilir. Bu katsayı kalibre edilebilmekle birlikte Cocomo II.2000 için değeri 0.91 olarak kabul edilmiştir. Mimari sonrası modelindeki başlangıç tahminlerini ayarlamak için kullanılan özellikler ise ürün, bilgisayar, personel ve proje olmak üzere dört sınıfa karşılık gelirler.

Ürün özellikleri: Geliştirilen yazılım ürününün gerekli özellikleri ile ilgilidir (Çizelge 2.9).

Çizelge 2.9 COCOMO II Ürün özellikleri

Ölçüt	**Açıklama**
RELY	Sistem güvenilirliği
DATA	Test veri tabanının büyüklüğü
CPLX	Modüllerinin karmaşıklığı
DOCU	Gerekli belgelenmenin büyüklüğü
RUSE	Yeniden kullanılabilir bileşenlerin oranı

Bilgisayar özellikleri: Donanım platformunun yazılıma verdiği kısıtlardır (Çizelge 2.10).

Çizelge 2.10 COCOMO II Bilgisayar özellikleri

Ölçüt	**Açıklama**
TIME	Çalışma zamanı kısıtları
STOR	Bellek kısıtları
PVOL	Platformun uçuculuğu

Proje özellikleri:

Proje özellikleri (Çizelge 2.11), yazılım geliştirme projesinin belirli özellikleri ile ilgilidir.

Çizelge 2.11 COCOMO II Proje özellikleri

Ölçüt	**Açıklama**
TOOL	Kullanılan yazılım geliştirme araçlarının gelişmişliği
SITE	Çalışanların birbirine yakın çalışma imkanı
SCED	Zaman sıkışıklığı

Personel özellikleri: Projede çalışanların deneyimlerini ve yeteneklerini hesaba katan çarpanlardır (Çizelge 2.12).

Çizelge 2.12 COCOMO II Personel özellikleri

Ölçüt	**Açıklama**
PCAP	Programcı yeteneği
ACAP	Proje analistinin yeteneği
PCON	Personel devamlılığı
APEX	Ekibin benzer uygulama deneyimi
PLEX	Ekibin benzer platform deneyimi
LTEX	Dil ve araç deneyimi

2.1.3 Öğrenme Tabanlı Modeller

Özellikle son 10 yılda bu alanda mevcut model ve tekniklerden de faydalanılarak daha iyi sonuçlar elde edilmeye yönelik yapılan çalışmaların sayısı artmıştır. “Yapay Sinir Ağları”, “Genetik Programlama”, “Bulanık Mantık”, “Karar Ağacı” gibi teknikler kullanılarak bu alanda yapılan çalışmalar bir adım daha ileriye taşınmaya çalışılmıştır.

2007 yılında yapılan bir çalışmada [18] 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 kullanılılarak bir model geliştirilmiş, sonuçlar regresyon tabanlı ve yapay sinir ağları yöntemleriyle kıyaslanmıştır. Sonuçların birbirine yakın olduğu görülmüştür[19].

Yazılım sektörünün gelişimine paralel olarak Türkiye’de de bu alanda çalışmalar yapılmaya başlanmıştır. 2006 yılında yapılan bir çalışmada [20] 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[21]. 2008 yılında yapılan başka bir çalışmada [22], ele alınan projeler için; çalışan kişi sayısı, yazılım büyüklüğü,

yazılım kod satır sayısı, bozunma oranı ve yazılım maliyeti gibi verileri toplanarak, bu veriler üzerinden yapay sinir ağları ile bir kestirim yapılmaya çalışılmış ve sonuçlar COCOMO II modeli ile karşılaştırılmıştır.

2.2 Karşılaştırma

Yazılım maliyet kestirim yöntemleri üç temel kategori altında gruplandırılabilir: Algoritmik Modeller, Algoritmik Olmayan Modeller ve Öğrenme Tabanlı Modeller. Bu modellerin birbirlerine göre güçlü ve zayıf yanları Çizelge 2.13'te gösterilmiştir.

Çizelge 2.13 Modellerin Karşılaştırması[12]

	Metot	Güçlü Yanları	Zayıf Yanları
Algoritmik Olmayan	Uzman Görüşü	İyi bir uzman bulunabilirse göreceli olarak iyi bir tahminde bulunabilir. Hızlıdır	Uzmana göre değişir. Tutarlı olamayabilir.
	Benzeşim	Gerçek projeler ve geçmiş tecrübe temeli üzerine kurulur	Benzer proje olmayabilir. Tarihsel veri doğru olmayabilir
	Parkinson Kazanmak için Fiyat	Kontratı genellikle kazanır	Çoğu zaman doğru bir tahmin yapılamaz.
	Yukardan Aşağıya	Sistem seviyesinde odaklanma sağlar. Aşağıdan Yukarıya yöntemine göre daha kolay ve daha hızlıdır. Çok az detay gerektirir.	Düzenlemeler konusunda az detay vardır. Diğer metotlara nazaran tutarlılığı azdır
	Aşağıdan Yukarıya	Detaylı analize ihtiyaç duyar. Proje izlemede diğerlerine göre avantaj ve alt seviye maliyetleri bulmakta olanak sağlar.	Maliyet faktörlerine daha fazla bakmak gerektirir. Daha fazla çaba gerektirir. Başlarda kestirim yapmak zordur
Algoritmik		Objektif ve tekrarlanabilir sonuçlar elde edilebilir. Kestirim yöntemini anlamak ve açıklamak daha kolaydır.	Girdiler objektif değildir. Geçmiş projelerin kullanılması mevcut şartlara uygun olmayabilir. Genellemeler yapmak zordur
Öğrenme Tabanlı		Geleneksel yöntemlere göre daha iyi sonuçlar elde edilebilir. Sürekli iyileştirmeye açıktırlar.	Çok miktarda geçmiş veriye ihtiyaç duyarlar.

2.3 Değerlendirme

Jorgensen ve Shepperd 2007 yılında yazılım maliyet kestirimi alanında o zamana kadar yapılan akademik çalışmaları incelemiştir[23]. Bu çalışmada 76 ayrı dergide yer alan 304 makale incelenmiştir. Sonuçlar Çizelge 2.14'de gösterilmiştir.

Çizelge 2.14 Çaba Kestirim Yöntemleri Üzerinde Yapılan Çalışmalar

Yöntem	**-1989**	**1990-1999**	**2000-2004**	**Genel**
Rg	21(51%)	76(47%)	51(51%)	148(49%)
An	1(2%)	15(9%)	15(15%)	31(10%)
Ej	3(7%)	22(13%)	21(21%)	46(15%)
Wb	3(7%)	5(3%)	4(4%)	12(4%)
Fp	7(17%)	47(29%)	14(14%)	68(22%)
Ct	0(0%)	5(3%)	9(9%)	14(5%)
Si	2(5%)	4(2%)	4(4%)	10(13%)
Nn	0(0%)	11(7%)	11(11%)	22(7%)
Th	20(49%)	14(9%)	5(5%)	39(13%)
By	0(0%)	1(1%)	6(6%)	7(2%)
Cb	0(0%)	3(2%)	2(2%)	5(2%)
Ot	2(5%)	7(4%)	16(6%)	25(8%)

Rg=Regression An=Analogy Ej=Expert Judgment Wb=Work Break Down Fp=Function Point Ct=Classification and Regression trees Si=Simulation Nn=Neural Network Th=Theory By=Bayesian Cb=Combination of estimates Ot=Other

Bu çalışmanın sonucuna göre regresyon tabanlı yaklaşımların (Rg) 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 (Fp), üçüncü sırada ise %15 ile uzman görüşüne (Ej) dayalı çalışmalar yer almaktadır.

Bu çalışmanın sonuçlarına göre işlev puanı ile ilgili yapılan çalışmalar 1990 larda zirveye ulaşmış daha sonra azalmaya başlamıştır. Putnam SLIM [5] , Halstead [24] gibi teori tabanlı kestirim yöntemleri (Th) 80'li yıllarda çok popüler olmakla birlikte, 1990'lardan sonra popülaritesini kaybetmişlerdir.

2000-2004 yılları arasında Diğer (Ot) kategorisi altındaki çalışmaların ivme kazandığı görülmektedir. Bu kategori Yapay Sinir Ağları, Genetik Programlama, Bulanık Mantık, Lineer Programlama gibi teknikleri içermektedir. Son yıllarda bu alanda yapılan çalışmaların ivmesinin artarak devam ettiği gözlemlenmektedir.

Yukarıda belirtilen modelleri değerlendirdiğimizde aşağıdaki sonuçlar göze çarpmaktadır:

- Hiç biri tüm projelere uygun olamıyor. İçinde bulunulan şartlara ve eldeki verilere göre uygun modelin seçilmesi gerekiyor.
- Bu yöntemlerin 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 sunar.
- Öğrenme tabanlı çalışmalar özellikle son 15 yılda ivme kazanmış ve kayda değer başarılı sonuçlar elde edilmiştir. Bu alanda daha fazla çalışma yapılmasına ihtiyaç vardır.

BÖLÜM 3

ÖNERİLEN MODEL

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 [25] 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 [26] veri tabanı bağlantılı 4GL uygulamalar için bir kestirim modeli sunmuştur. Bu model aşağıdaki kabuller doğrultusunda geliştirilmiştir:

1. 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
2. Sistem veritabanı ER diyagramından otomatik olarak üretilir. Bundan dolayı geliştirme çabası açısından ER diyagramları arasındaki karmaşıklık farkı ihmal edilebilir.
3. 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 ayrıca veritabanı sistemlerine yönelik Bayes tabanlı istatistiğe dayanan bir çaba kestirim modeli geliştirmiştir[27]. Bu model önceki modelin istatistiksel yaklaşımla geliştirilmiş bir türevidir. Riquelme [28] 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 [29] bir proje için gereken çabayı hesaplayan bir model geliştirmiştir. Buna göre toplam çaba eşitlik 3.1 de belirtilmiştir.

$$LOE_p = a + b_1 \text{ x Formlar} + b_2 \text{ x Raporlar} + b_3 \text{ x Tablolar} + b_4 \text{ x Modüller} \quad (3.1)$$

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.

Tez kapsamında yapılan çalışmada yukarıda bahsedilen modeller incelenerek her modelin pozitif tarafları alınmış eksik kalan tarafları üzerine yoğunlaşılarak yeni bir model geliştirilmiştir. Bu çalışmada üzerinde çokça çalışılmış ve geniş kabul görmüş bir model olan COCOMO II modelinin tecrübesinden de yararlanılarak form tabanlı uygulamalar için yeni bir model 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. Örneğin 2010 yılında yapılan bir çalışmada COCOMO modeli, yapay sinir ağları ile birlikte kullanılarak daha hassas kestirimler yapılmıştır [31]. 2011 yılında yapılan başka bir çalışmada bulanık mantık kullanılarak FL-COCOMO II isminde yeni bir model geliştirilmiş ve klasik COCOMO II modeline göre daha başarılı sonuçlar elde edilmiştir[32]. 2014 yılında yapılan bir çalışmada ise yapay sinir ağları kullanılarak karma bir model oluşturulmaya çalışılmıştır[33]. 2010 yılında yapılan bir çalışmada savunma sanayi projeleri üzerinde kalibrasyon yapılarak daha başarılı sonuçlar elde edildiği görülmüştür[34].

2006 yılında yapılan başka bir çalışmada ise NASA proje verileri kullanılarak genetik algoritmalar yolu ile COCOMO benzeri bir model ortaya konmuştur [35].

- 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.

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. Çizelge 3.1'de verilmiş olan bu matrisin oluşumunda uzman görüşüne başvurularak, uzmanların geçmiş tecrübelerine göre elde edilen satır sayıları minimum ve maksimum değerlerin oluşturulmasında kullanılmıştır. Çizelgedeki ara değerler ise maksimum ve minimum değerler arasındaki farkın eşit dağılımı şeklinde hesaplanmıştır.

Çizelge 3.1 Nesne/Satır Sayısı Dönüşüm Matrisi

		Karmaşıklık Dereceleri									
		1	2	3	4	5	6	7	8	9	10
Nesne Türleri	Form	10	56	103	150	197	244	291	338	385	430
	Rapor	10	58	107	156	205	254	303	352	401	450
	Prosedür	50	116	183	249	316	383	449	516	583	650
	Enteg-rasyon	10	44	78	113	147	182	216	251	285	320
	Tablo	5	16	27	38	49	60	71	82	93	105

Form: Birden fazla bileşenden oluşan bir ara yüzdür. Sürükle bırak teknolojisiyle hazır bileşenler kullanılarak kod yazmadan üretilebildiği gibi özel ihtiyaçlar için kod geliştirmek

gerekebilir. Basit bir forma örnek giriş ekranı olabilir. Karmaşık bir form olarak giriş, güncelleme, silme işlemleri yapılan, master/detay grid yapısı olan bir form örnek olarak gösterilebilir.

Rapor: Gelişmiş raporlama araçları kullanılarak kod yazmadan rapor oluşturulabileceği gibi gelişmiş arama ve filtreleme ihtiyaçlarına bağlı olarak karmaşık ve dinamik SQL sorgularının ve gömülü prosedürlerin oluşturulmasını gerektiren raporlarlarda geliştirilebilir.

Tablo: Veritabanı seviyesinde verilerin saklanmasına yarayan nesnelerdir. Tablolar hazır araçlar sayesinde kod yazmadan oluşturulabilir. Fakat bu da geliştirici için bir efor anlamına gelecektir. Tablonun alan sayısının fazlalığı, constraint, index gibi nesnelerin tanımlanmasına göre harcanması gereken efor artacaktır.

Prosedür: Görsel arayüzler ile ifade edilemeyen bir fonksiyonu ifa etmek için geliştirilen kodu ifade eder. Birtakım hesaplama algoritmaları, proses ve olay yönetimleri, toplu işlemler gibi bir girdisi ve çıktısı olan kod parçaları bu sınıfa girer.

Entegrasyon: İki farklı sistem arasında veri transferi ya da iletişim amacıyla geliştirilen kodu ifade eder. Günümüzde bu iletişim genellikle senkron olarak web servisler ile sağlanmakla birlikte dosya transferi şeklinde asenkron şekilde de yapılabilmektedir.

Bir form tabanlı uygulama için büyüklük, form içerisinde kullanılan nesneler ile modüllerin satır sayısına dönüştürülmüş toplamları ile ifade edilebilir. Bunu ifade eden formül (3.2) de verilmiştir.

$$TB=\sum_{i=1}^{n} F_i x S_i + \sum_{i=1}^{n} R_i x S_i + \sum_{i=1}^{n} T_i x S_i + \sum_{i=1}^{n} E_i x S_i + \sum_{i=1}^{n} P_i x S_i \qquad (3.2)$$

Bu formülde her nesnenin karmaşıklık derecesine göre farklı satır sayısı Çizelge 3.1'te belirtilen değerler doğrultusunda elde edilerek hesaplanacaktır. Örneğin uygulama içerisinde 5 adet çok karmaşık form geliştirilecekse bunların satır sayısı F=5*430=2.150 olacaktır.

Bu çalışmada Çizelge 3.2'de yer alan Cocomo II.2000 Mimari Sonrası Evre için kalibre edilmiş ölçüt seti ve değerleri kullanılmıştır.

Çizelge 3.2 Cocomo II.2000 Mimari Sonrası Evre Ölçüt Değerleri

Ölçüt	Çok Düşük	Düşük	Normal	Yüksek	Çok Yüksek	Aşırı Yüksek
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
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	

3.1 Yerel Şartlara Göre Model Kalibrasyonu

Yapılan çalışmalar COCOMO modelinin yerel şartlara göre kalibre edilmesi ile daha başarılı sonuçlar elde edildiğini ortaya koymuştur. Bu çalışmada bu husus göz önüne alınarak ana formülde (2.9) yer alan A katsayısının kalibre edilmesi düşünülmüştür. Kalibrasyon için birden fazla teknik vardır. Bu çalışmada doğal logaritma tekniği kullanılmıştır.

3.1.1 Verilerin Toplanması

Kalibrasyon yapılabilmesi için veriye ihtiyaç vardır. Bunun için Çizelge 3.3'de yer alan, telekomünikasyon sektöründe faaliyet gösteren bir firmanın, Java ve C# dillerinde geliştirilen 15 adet proje verisi kullanılmıştır. Bu projelerin 13 tanesi dış kaynak, 2 tanesi ise iç kaynaklarla geliştirilmiştir. Projeler geliştirilmeye başlanmadan önce, çaba kestirimi yapabilmek adına ihtiyaç duyulan verilerin toplanabilmesi için, Şekil 3.8'de ekran görüntüsü verilen uygulamaya benzer bir uygulama geliştirilmiştir. Bu uygulamaya, her bir proje için Çizelge 3.2'de belirtilen çaba çarpanları, ölçek faktörleri, geliştirilecek olan nesne sayıları ve bunların karmaşıklık değerleri girilmiştir. Bu değerlerin hepsi kurum içinde tutarlılık denetiminden geçirilerek kullanılmıştır. Bu değerlere göre tahmini adam gün süreleri hesaplanmıştır. Hesaplanan bu değerler Çizelge 3.3'deki tabloda üçüncü sütunda 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 Çizelge 3.3'deki tabloda ikinci sütunda gösterilmiştir.

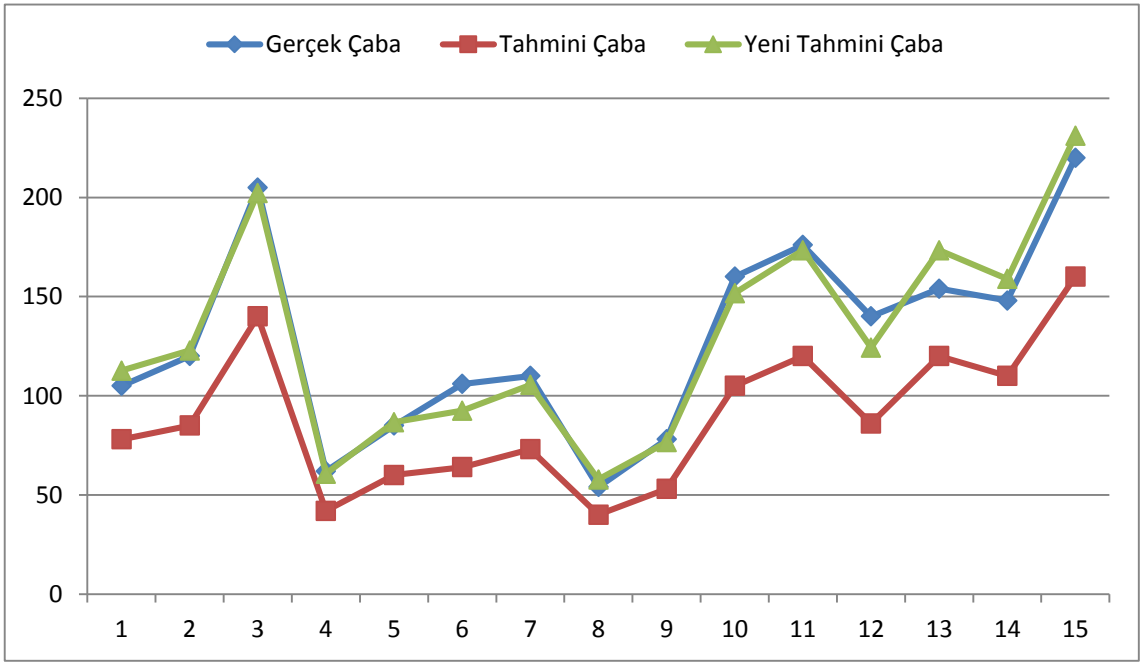
3.1.2 Kalibrasyon

Bu çalışma altı aylık bir zaman dilimi içerisinde yapılmıştır. Zaman kısıtından dolayı farklı dillerle ve farklı alanlarda geliştirilen proje verileri kullanılmıştır. Kalibrasyonun daha homojen verilerle yapılmasının daha doğru sonuçlar vereceği düşünülmektedir.

Kalibrasyon için, gerçekleşen çaba değerinin logaritması ile formülde yer alan A katsayısı olmadan hesaplanan çaba değerinin logaritması ($Büyüklük^E$ * M) arasındaki farklar hesaplanmış, daha sonra bu farkların ortalaması alınarak bir X değeri hesaplanmıştır. Bu X değerinin ters logaritması alınarak 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 Çizelge 3.3'de yedinci 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 3.1).

Çizelge 3.3 Kalibrasyon Tablosu

Proje No	Gerçek Çaba	Tahmini Çaba	ln (Gerçek Çaba)	Ln (Tahmini Çaba/2,94)	Fark	Yeni Tahmini Çaba
1	105	78	4,653960	3,278299	1,375661	112,69
2	120	85	4,787492	3,364242	1,423250	122,80
3	205	140	5,323010	3,863233	1,459777	202,26
4	62	42	4,127134	2,659260	1,467874	60,68
5	85	60	4,442651	3,015935	1,426716	86,68
6	106	64	4,663439	3,080474	1,582966	92,46
7	110	73	4,700480	3,212050	1,488431	105,46
8	54	40	3,988984	2,610470	1,378514	57,79
9	78	53	4,356709	2,891882	1,464826	76,57
10	160	105	5,075174	3,575551	1,499623	151,69
11	176	120	5,170484	3,709082	1,461402	173,36
12	140	86	4,941642	3,375938	1,565705	124,24
13	154	120	5,036953	3,709082	1,327870	173,36
14	148	110	4,997212	3,622071	1,375141	158,92
15	220	160	5,393628	3,996764	1,396863	231,15



Şekil 3.1 Çabaların Karşılaştırılması

Çizelge 3.4'te ise, 15 adet proje için, kalibrasyon yapıldıktan sonra tahmin edilen çaba ile gerçekleşen çabaların karşılaştırılması ve hata oranları gösterilmiştir. Hata oranı (MRE) eşitlik 3.3'e göre hesaplanmıştır. Bu çizelgede yer alan hata oranlarının ortalamasının (MMRE) 5,59 olduğu görülmektedir: Hata oranlarının mutlak değerlerinin ortalaması 5,59 etmektedir.

MRE=100 x (Gerçek Çaba-Tahmini Çaba)/Gerçek Çaba (3.3)

Çizelge 3.4 Hata Oranı Tablosu

Proje 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

3.2 Kalibre Edilmiş Model Kullanılarak Kestirim Örneği

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. Ekran prototipleri Şekil 3.2, 3.3, 3.4, 3.5, 3.6 ve 3.7'de gösterilmiştir. Form, entegrasyon, tablo sayıları ve karmaşıklık değerleri Çizelge 3.5 de gösterilmiştir.

Çizelge 3.5 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. Seçilen Ölçek Faktörleri Çizelge 3.6'de, Çaba Çarpanları Çizelge 3.7'de gösterilmiştir.

Çizelge 3.6 Örnek Proje Ölçek Faktörleri

PREC	3,72
FLEX	3,04
RESL	4,24
TEAM	2,19
PMAT	4,68

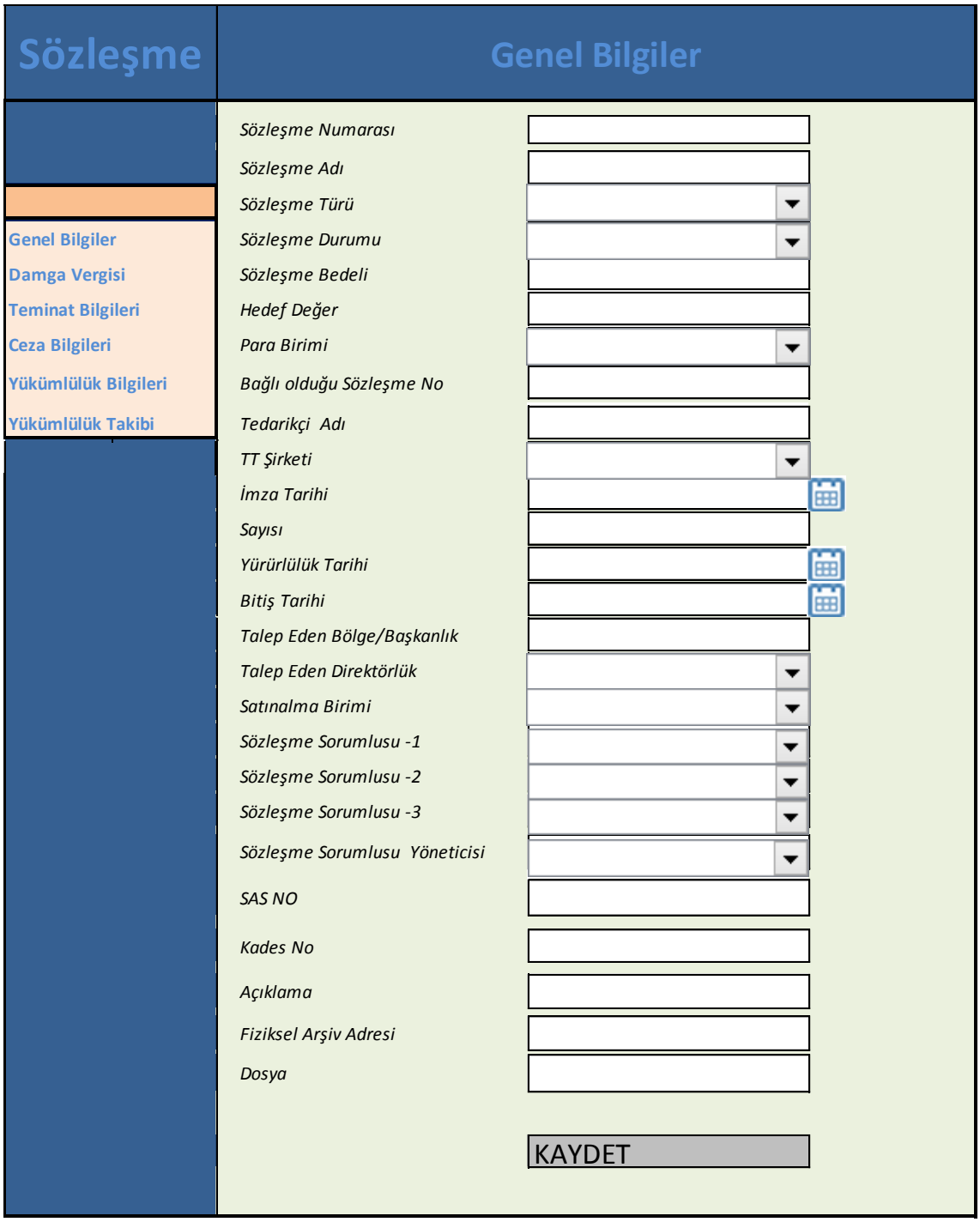
Çizelge 3.7 Örnek Proje Çaba Çarpanları

RELY	1
DATA	2
CPLX	1
RUSE	1
DOCU	1
ACAP	1
PCAP	1
PCON	1
APEX	0,88
PLEX	0,91
LTEX	1
TOOL	0,78
SITE	0,86
SCED	1

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.

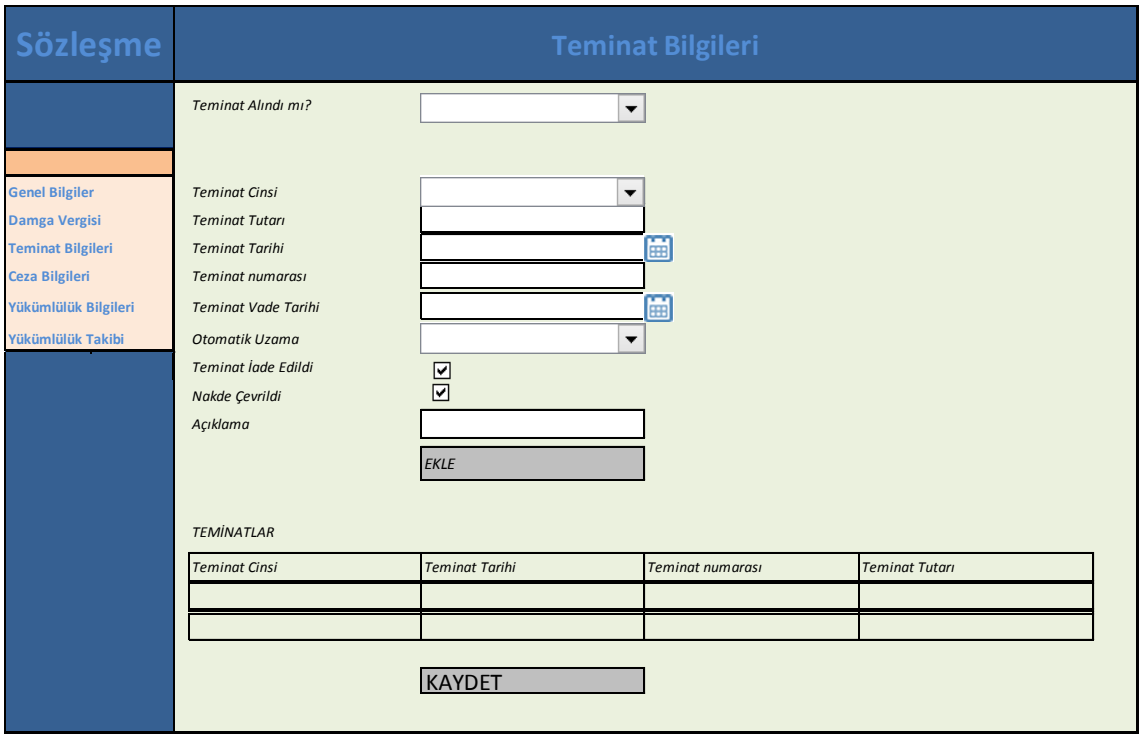
$TB = \sum_{i=1}^{n} F_i x S_i + \sum_{i=1}^{n} R_i x S_i + \sum_{i=1}^{n} T_i x S_i + \sum_{i=1}^{n} E_i x S_i + \sum_{i=1}^{n} P_i x S_i$ formülüyle ifade edilen toplam büyüklük (Toplam Satır Sayısı) =1,413; 14 adet Çaba Çarpanının çarpım değeri $\prod_{1}^{14} Çaba\ Çarbanı$=0,537176 olmaktadır. Bu değerlere göre (2.9) 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 çalışmada, yukarıda belirtilen verilerin girilebilmesi ve hesaplamaların yapılabilmesi için C# dilinde bir masa üstü uygulaması da gerçeklenmiştir. Programın ekran görüntüleri Şekil 3.8, 3.9, 3.10, 3.11 ve 3.12 de gösterilmiştir.

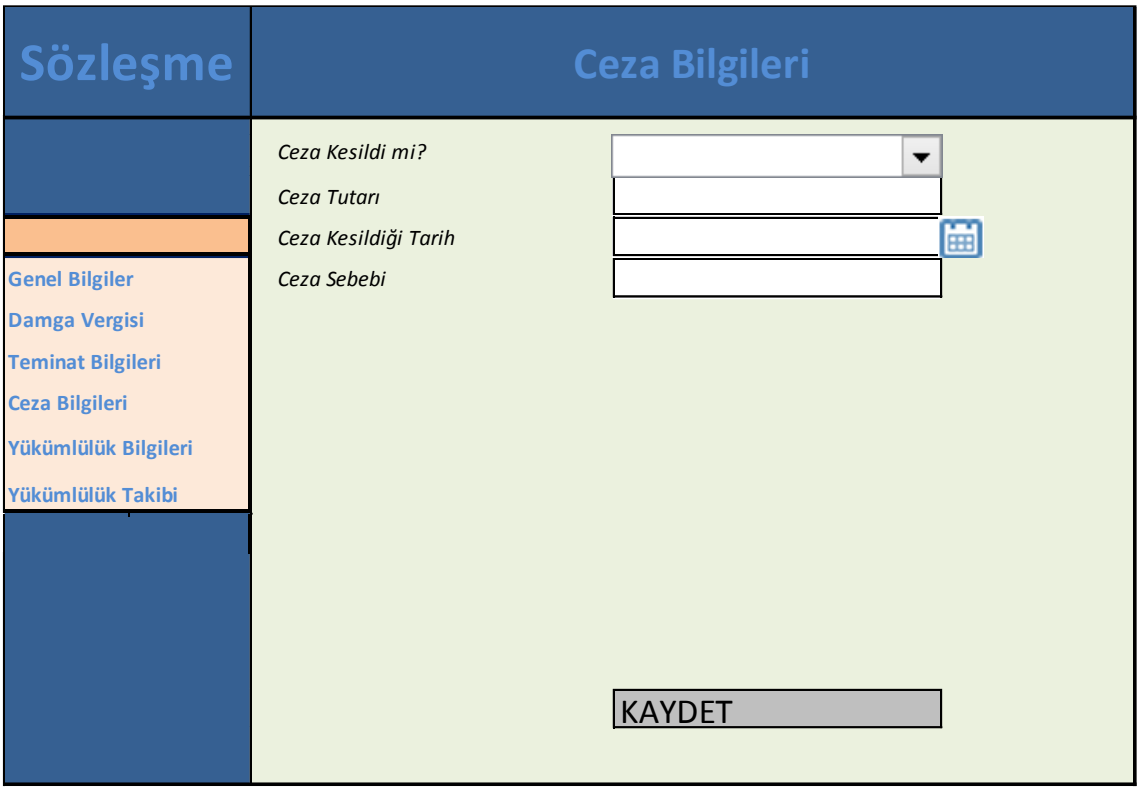


Şekil 3.2 Sözleşme Yönetimi Genel Bilgiler Ekranı

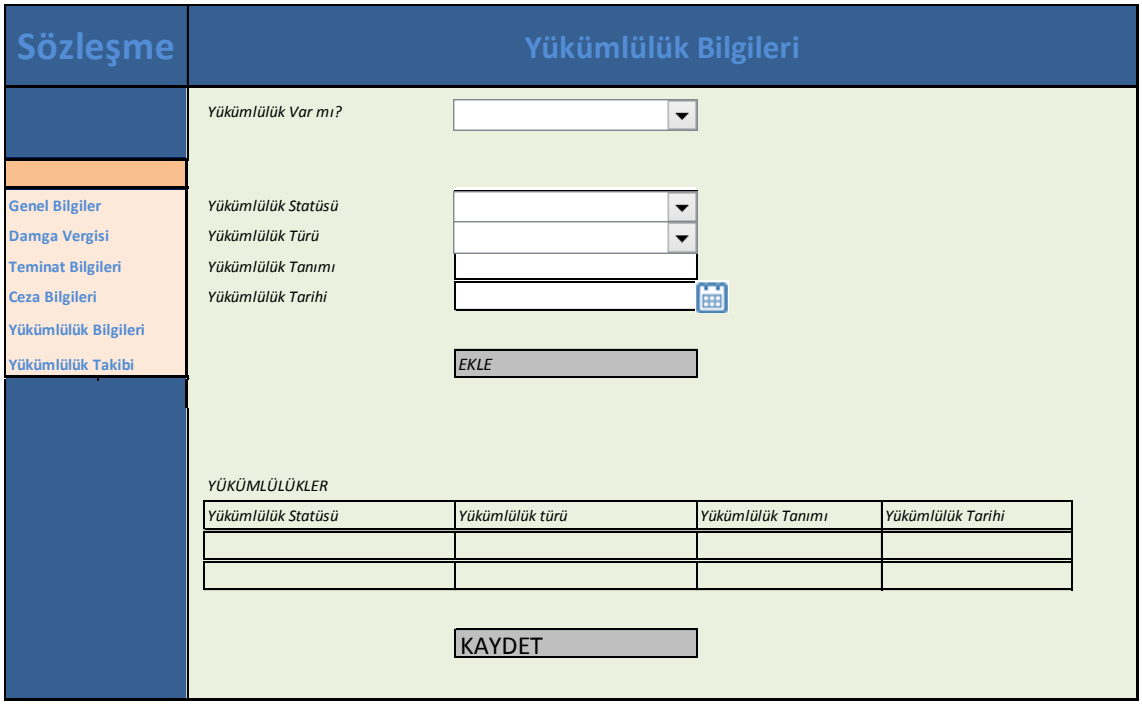
Şekil 3.3 Sözleşme Yönetimi Damga Vergisi Ekranı



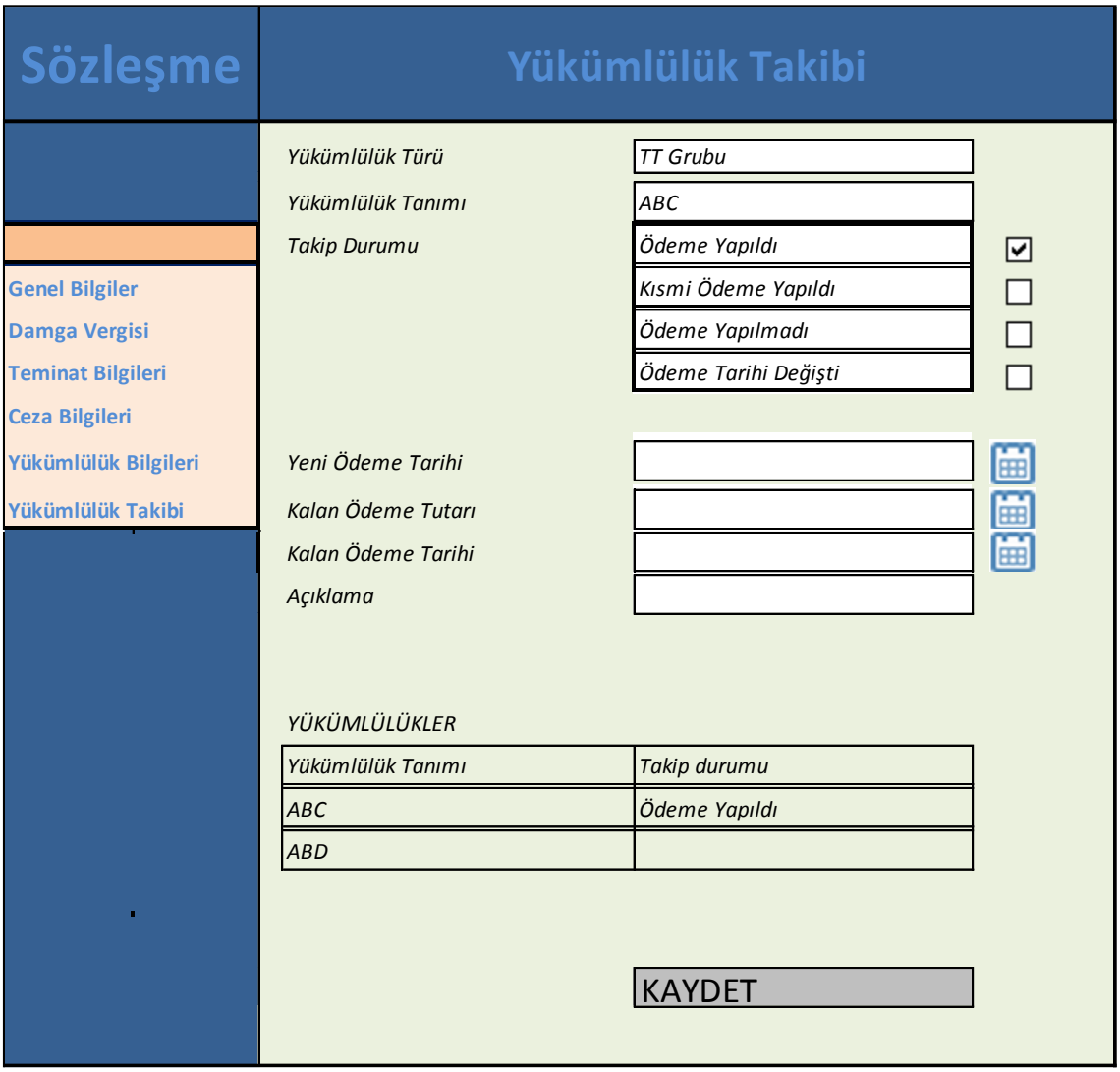
Şekil 3.4 Sözleşme Yönetimi Teminat Bilgileri



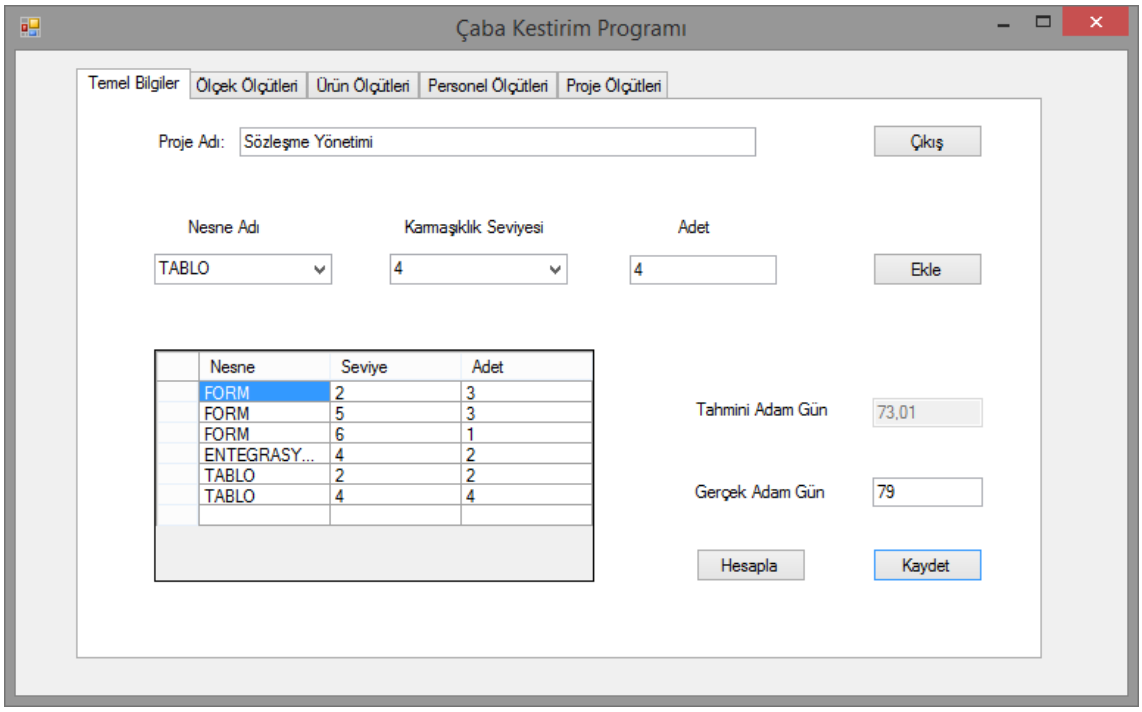
Şekil 3.5 Sözleşme Yönetimi Ceza Bilgileri Ekranı



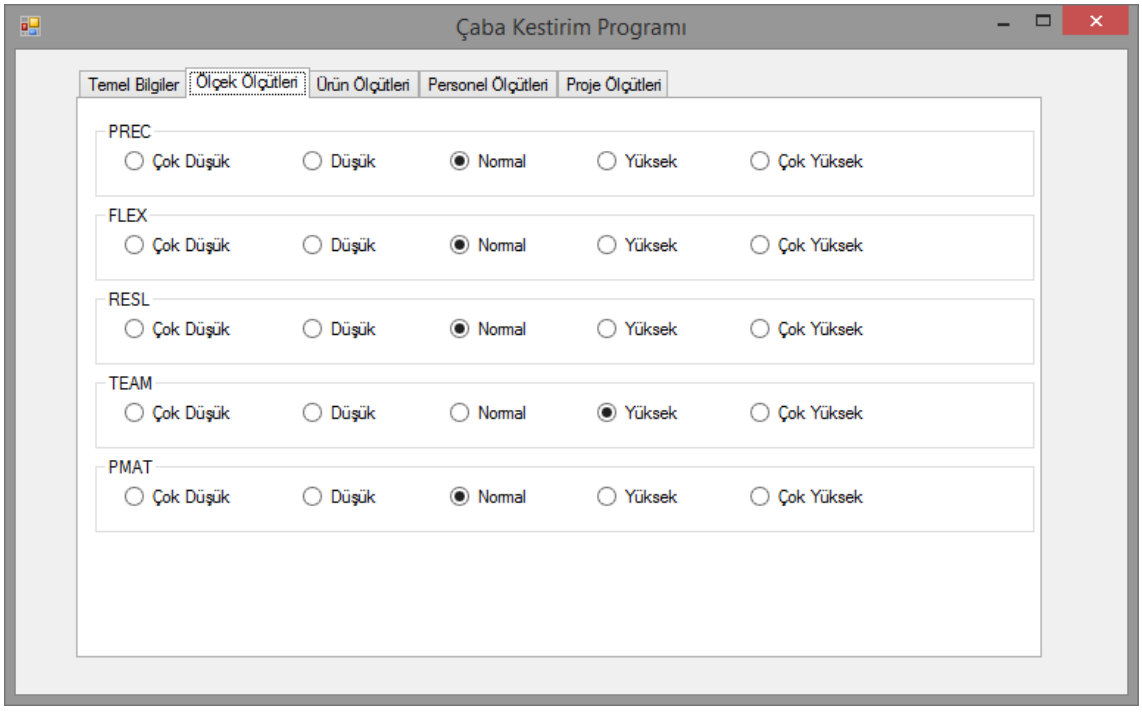
Şekil 3.6 Sözleşme Yönetimi Yükümlülük Bilgileri Ekranı



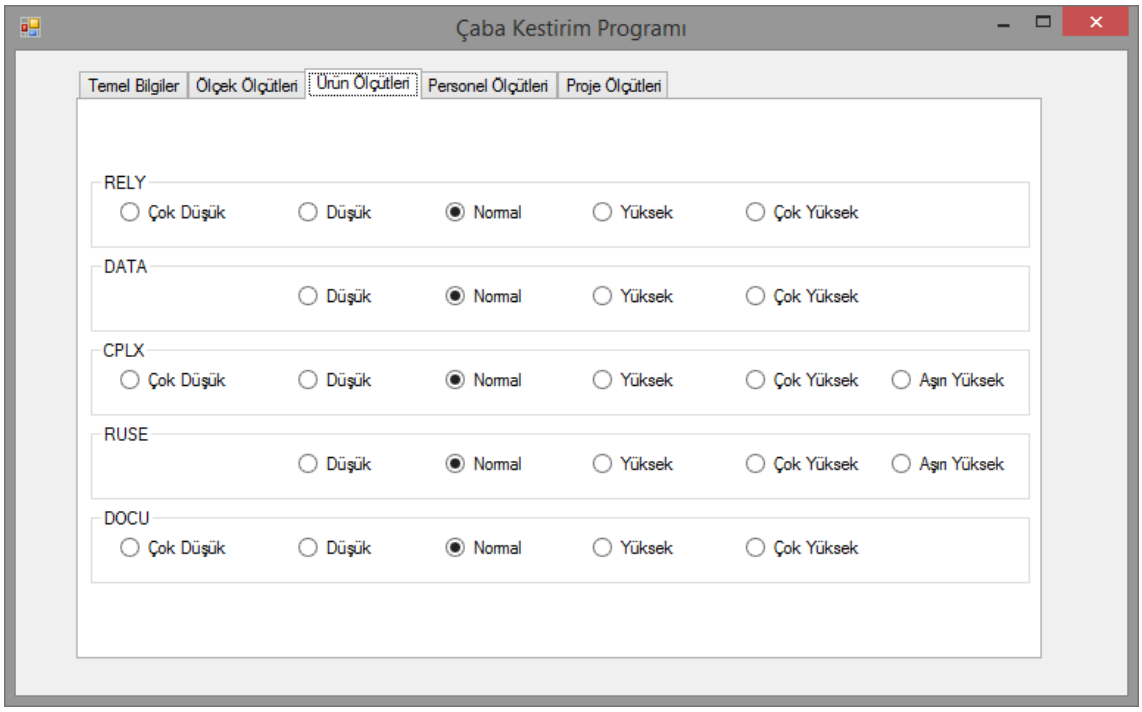
Şekil 3.7 Sözleşme Yönetimi Yükümlülük Takibi Ekranı



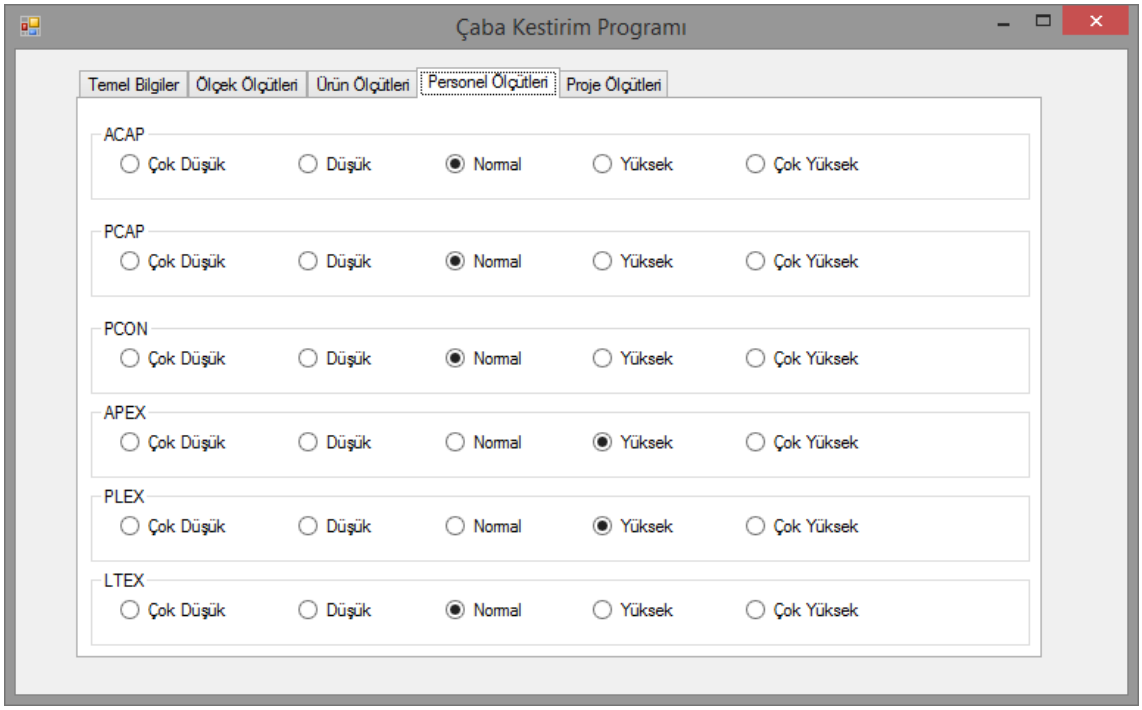
Şekil 3.8 Çaba Kestirim Programı-Temel Bilgiler



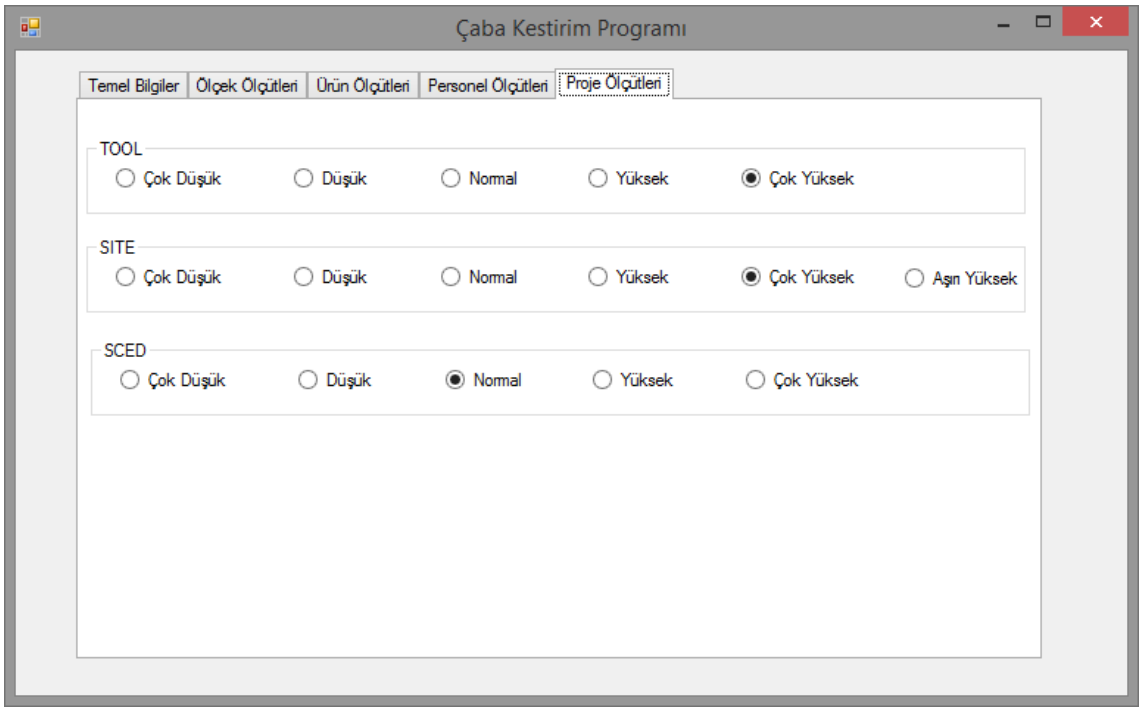
Şekil 3.9 Çaba Kestirim Programı-Ölçek Ölçütleri



Şekil 3.10 Çaba Kestirim Programı- Ürün Ölçütleri



Şekil 3.11 Çaba Kestirim Programı- Personel Ölçütleri



Şekil 3.12 Çaba Kestirim Programı- Proje Ölçütleri

BÖLÜM 4

SONUÇ VE ÖNERİLER

Bu çalışmada çaba kestirim yöntemleri arasında en çok kullanılan COCOMO II.2000 modelinden faydalanılarak, günümüzde yaygın olarak kullanılan form tabanlı uygulamalar için bir çaba kestirim modeli oluşturulmaya çalışılmıştır. Ölçüt kümesi içerisinde yer alan bilgisayar özellikleri TIME, STOR, PVOL günümüz şartlarında geçerli ölçütler olarak değerlendirilmediğinden 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 [30].

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 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.

Bu çalışma günümüz dünyasında sıkça kullanılan form tabanlı uygulamalar için çaba kestirimine yönelik yeni bir yaklaşım sunan, pratikte kullanılabilen ve bundan sonra yapılacak olan benzer çalışmalar için yönlendirici bir çalışma olmuştur.

KAYNAKLAR

[1] The Standish Group. (1995). CHAOS Chronicles, Standish Group Int. Report., USA

[2] The Standish Group. (2013). CHAOS Manifesto, Standish Group Int. Report., USA

[3] McCabe, T.J. (1976). "A Complexity Measure," Software Engineering, IEEE Transactions on, 2(4)

[4] Tom Gilb. (1977). “Software Metrics” (ISBN 0876268556)

[5] Putman W.L.. (1978). "A general empirical solution to the macro software sizing and estimating problem.", IEEE Trans. on Softw. Eng., 4(4).

[6] Albrecht, A. J. (1979). "Measuring application development productivity", Proc. IBM Applications Develop. Symp.

[7] Boehm, B.W., (1981). “Software Engineering Economics”, Prentice Hall.

[8] Londeix. (1987). “Cost Estimation for Software Development”, Addison Wesley, Reading, Massachusetts

[9] Boehm, B., Bradford, C., Horowitz, E., Madachy, R., Shelby, R. ve Westland C. (1995). "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0", Annals of Software Engineering Special Volume, Amsterdam, Netherlands.

[10] Fenton, N.E. ve Pfleeger S. (1998). "Software Metrics: A Rigorous and Practical Approach", International Thomson Computer Press.

[11] Boehm, B., C. Abts ve S. Chulani. (2000). “Software Development Cost Estimation Approaches – A Survey”, Annals of Software Engineering, 10(1-4): 177-205.

[12] Leung H. ve Fan Z. (2002). “In Handbook of Software Engineering and Knowledge Engineering”

[13] Caper Jones. (2007). “Estimating software cost”, Tata Mc- Graw -Hill Edition

[14] Murali Chemuturi. (2009). “Analogy based Software Estimation” Chemuturi Consultants

[15] Pressman, R.S. (2005). “Software Engineering A Practitioner’s Approach”, 6th Edition, McGraw-Hill International.

[16] Sommerville I. (1992). "Software Engineering", Addison Wesley Publishing Company, Workingham, England.

[17] Selby, R.W. (1988). “Empirically analyzing software Reuse in a production environment”

[18] Kanmani, S., Kathiravan, J., Senthil Kumar, S., ve Shanmugam, M. (2007). “Neural Network Based Effort Estimation using Class Points for OO Systems In Computing: Theory and Applications”, ICCTA'07. International Conference

[19] Chavoya, A., Lopez-Martin, C. ve Meda-Campa, M. E. (2011). “Applying Genetic Programming for Estimating Software Development Effort of Short scale Projects ”, In Information Technology: New Generations (ITNG), Eighth International Conference

[20] Baskeles, B., Turhan, B. ve Bener, A. (2007). "Software effort estimation using machine learning methods,” Computer and information sciences.

[21] Ayyıldız,M., Kalıpsız, O. ve Yavuz, S. (2008). “Metric-Set and Model Suggestion for Better Software Project Cost Estimation.” International Journal of Computer, Information, Systems and Control Engineering.

[22] Sezer A. ve Okatan, A. (2008). “Yazılım projelerinde yapay sinir ağı uygulaması ile maliyet tahmini”

[23] Jorgensen, M. ve Shepperd, M. (2007). “A Systematic Review of Software Development Cost Estimation Studies”, Software Engineering, IEEE Transactions on 33(1)

[24] Halstead M. H. (1977). “Elements of Software Science”

[25] Smith, R.K. (1998). “Effort estimation in component based software development: identifying parameters”, 29th ACM SIGCSE Technical Symp., Atlanta, GA

[26] Van Koten C. ve Grey A. (2009). “Bayesian statistical effort prediction models for data-centred 4GL sofware development”, Department of Information Science, University of Otago, Dunedin, New Zealand

[27] Van Koten C. (2004). “An effort prediction model for data-centred fourth-generation-language software development”, Department of Information Science, University of Otago, Dunedin, New Zealand

[28] Riquelme, J.C, Polo, M., Aguilar Ruiz, J.S., Piattini M., Ferrer Troyano, F. J. ve Ruiz, F. (2006). “A comparison of effort estimation methods for 4GL programs”

[29] Morgan Peeples J.N. (2006). “A software development cost estimation model for higher level languages environments”

[30] Zia Z., Abdur Rashid ve Khair uz Zaman. (2011). "Software cost estimation for component-based fourth-generation-language software applications."

[31] Attarzadeh, I. ve Siew Hock Ow. (2010). "Proposing a new software cost estimation model based on artificial neural networks," Computer Engineering and Technology (ICCET), 2010 2nd International Conference

[32] Attarzadeh, I. ve Siew Hock Ow. (2011). "Improving estimation accuracy of the COCOMO II using an adaptive fuzzy logic model," Fuzzy Systems (FUZZ), 2011 IEEE International Conference

[33] Patil L.V., Waghmode, R.M., Joshi, S.D. ve Khanna V. (2014). "Generic model of software cost estimation: A hybrid approach," Advance Computing Conference (IACC), 2014 IEEE International, 1379(1384): 21-22

[34] Taeho, L., Donoh, C. ve Jongmoon, B. (2010). "Empirical study on enhancing the accuracy of software cost estimation model for defense software development project applications" Advanced Communication Technology (ICACT), 2010 The 12th International Conference [

[35] Sheta, A. F. (2006). “Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects.”, Journal of Computer Science, 2(2): 118

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	:Uğur Kemal HAŞLAK
Doğum Tarihi ve Yeri	:17.01.1971
Yabancı Dili	:İngilizce
E-posta	:uhaslak@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Bil. Müh.	YTÜ	1994
Lise	Matematik	Tavas lisesi	1989

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2008-	Turk Telekom	Kurumsal Uygulamalar Müdürü
2005-2008	Finansbank	Kıdemli Yazılım Mimari
2001-2005	Denizbank	Takım Lideri
1999-2001	İhlas Finans	Kıdemli Yazılım Uzmanı
1995-1998	Takasbank	Yazılım Uzmanı
1994-1995	Belbim	Yazılım Uzmanı