

Yazılım Ürün Hatları için Otomatik İşlevsel Büyüklük Ölçümü Yaklaşımı

Önder Eren¹, Barış Özkan² ve Onur Demirörs³

¹ ASELSAN A.Ş. Savunma Sistem Teknolojileri Grubu, Ankara

² Bilişim Sistemleri Mühendisliği, Atılım Üniversitesi, Ankara

³ Enformatik Enstitüsü, Orta Doğu Teknik Üniversitesi, Ankara

¹oeren@aselsan.com.tr, ²baris.ozkan@atilim.edu.tr,
³demirors@metu.edu.tr

Öz. Yazılım büyüklük ölçümünü doğru yapmak, düşük maliyetli gömülü sistemler geliştirmek için çok önemlidir. Ancak; yazılım büyüklük ölçümü çok yakın zamana kadar büyük çoğunlukla manuel, zaman harcayan ve hataya açık bir süreç olmuştur. Bu zaman ve para kaybıyla sonuçlanabilir. Bu sürecin otomatize edilmesi yazılım geliştiren şirketlerde projenin kalitesini ve bütçe planlamasını iyileştirmek için bir zorunluluktur. Bu çalışmada bileşen tabanlı yazılım ürün hatlarında gerekli olan bilgiyi UML diyagramlarından alarak işlevsel büyüklüğü hızlı ve yaklaşık bir şekilde ölçebilmek için bir otomasyon yaklaşımı önerilmektedir. Çalışmada yazılım işlevsel büyüklüğü yaklaşımı için bileşen tabanlı yazılım ürün hatları ortamlarında kullanılan UML modelleri esas alınmış, COSMIC işlevsel büyüklük yönteminde önerilen işlevsellik öğeleri ile UML modelleri öğeleri arasında eşleştirme yapılmıştır. Yapılan bir durum çalışmasıyla, arayüz tabanlı tasarım ile hazırlanmış bileşen tabanlı ürün hattında manuel olarak ölçülen işlevsel büyüklüğün tahmin edilebildiği gösterilmiştir. Bu çalışma tamamlandığında, yazılım işlevsel büyüklük ölçümü için gerekli olan zaman ve iş gücünün azaltılması, ölçümün doğruluğunun artırılması ve geriye dönük işlevsel büyüklük hesaplamaları mümkün olacaktır.

Anahtar kelimeler: İşlevsel Büyüklük Ölçümü, Yazılım Ürün Hattı, Arayüz Tabanlı Tasarım

1 Giriş

Ölçme biliminin başlangıç noktası ve mühendisliğin en temel parçası olduğundan dolayı bir projenin bitirilebilmesi için öngörülebilir bulunabilmeye yardımcı olmaktadır. Yazılım büyüklük ölçümü kestirimi projenin tamamlanabilmesi için gerekli kaynakların belirlenmesinde önemli rol oynamaktadır. Proje yöneticileri yazılım yönetimi ve planlama yapabilmek için yazılımın tahmini büyüklüğünü mümkün

olduđu kadar hızlı ve yaklaşık bir şekilde bilmelidir[1]. Yazılım ürün hatları, deđişen müşteri isteklerine daha hızlı cevap verebilmek, daha önce yapılan işi tekrar yapmamak ve projeyi taahhüt edilen sürede tamamlamak için büyük ve yazılım odaklı projelerde kullanılır[2]. Yazılım büyüklüğü ölçümü bazı yazılım firmalarında göz ardı edildiđi için yazılım projeleri genel olarak ya zamanında tamamlanamamakta veya projenin başlangıcında belirlenen maliyet aşılmaktadır[3]. Yazılım büyüklük ölçümünün göz ardı edilme nedeni bu yöntemlerin otomatize edilmemesi nedeniyle zaman alıcı olması ve deneyim gerektirmesidir[4].

Bileşen tabanlı yazılım ürün hattı yaklaşımında bileşen havuzu oluşturularak bileşenlerin tekrar kullanılabilmesi amaçlanır[5]. Yazılım ürün hattı yaklaşımında genel olarak UML (Unified Modeling Language) tabanlı yazılım geliştirme araçları kullanılır[6]. Model güdümlü yazılım geliştirme araçları platform bağımsızlığı ve görsel bir kompozisyon sağladığı için günümüzde yaygın olarak tercih edilir[7]. UML araçları özellikle arayüz tabanlı tasarıma elverişli olması sebebiyle yazılım ürün hattında bulunan bileşenlerin soyutlanması ve farklı projelerde tekrar kullanılmasında kolaylık sağlar. Arayüz tabanlı bileşen tasarımında bileşenlerde kullanılacak arayüzler bileşen yazılmadan önce belirlenmiş olduğundan yazılım büyüklük ölçümü her bir bileşen için otomatik hale getirilerek yazılım süreci başlamadan bileşenin büyüklüğünün hesaplanması ve dolayısı ile proje yönetimi sürecinin kolaylaştırılması mümkün olmaktadır. Bu çalışmada arayüz tabanlı ürün hatları için UML diyagramları kullanılarak bu sürecin otomatize edilmesi ve bu sayede ölçümün daha hızlı yapılması ve ölçümü yapan kişinin deneyiminden bağımsızlığının sağlanarak objektif bir bakış açısı elde edilmesi hedeflenmektedir. Günümüzde yaygın olarak kabul görmüş olan COSMIC yazılım büyüklük ölçüm yöntemi temel alınarak bu yöntemin arayüz tabanlı ürün hattı tasarımlarında otomatize edilmesi amaçlanmaktadır. COSMIC yöntemi ile UML araçları kullanılarak yapılan arayüz tabanlı ürün hattı tasarımları arasında bir ilişki kurulması ve bu ilişkiden otomatik büyüklük ölçümü yapılması halinde yazılım büyüklük ölçümünde harcanan iş gücünün azaltılması ve tahminin doğruluğunun artırılması düşünülmektedir.

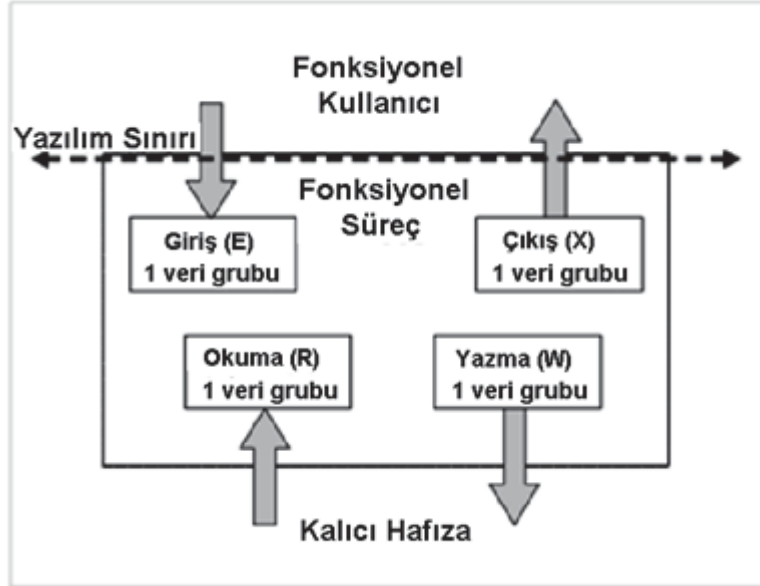
2 UML Modelleri ile İşlevsel Büyüklük Ölçümü

COSMIC İşlevsel Büyüklük Ölçüm Metodu

COSMIC metodu yaygın şekilde kullanılan ve diğer yazılım büyüklük ölçüm metodlarının platform bağımlılığı, kodlayan kişiye bağımlılık vb. zayıf yönlerini barındırmayan güncel ve yenilenmiş bir işlevsel büyüklük metodudur[8]. COSMIC yönteminde sistem sınırlarından giren her giriş ve sistem sınırlarından çıkan her çıkış 1 fonksiyon puanı alır ayrıca veri tabanına giriş ve veri tabanından çıkışlar da 1'er puan olarak hesaplanır. Şekil 1'de COSMIC yönteminde işlevsel büyüklük ölçümünde rol oynayan işlemler gösterilmiştir

- Giriş (E) bir veri grubunun fonksiyonel kullanıcıdan yazılım sınırlarından geçerek fonksiyonel süreçte kullanılacağı yere taşınması işlemidir.

- Çıkış (X) bir veri grubunun fonksiyonel süreçten yazılım sınırlarından geçerek fonksiyonel kullanıcıya taşınması işlemidir.



Şekil 1 - Veri Hareket Tipleri

- Okuma (R) bir veri grubunun kalıcı bellekten fonksiyonel süreçte kullanılacağı yere taşınması işlemidir.
- Yazma (W) bir veri grubunun fonksiyonel süreçten kalıcı belleğe taşınması işlemidir.

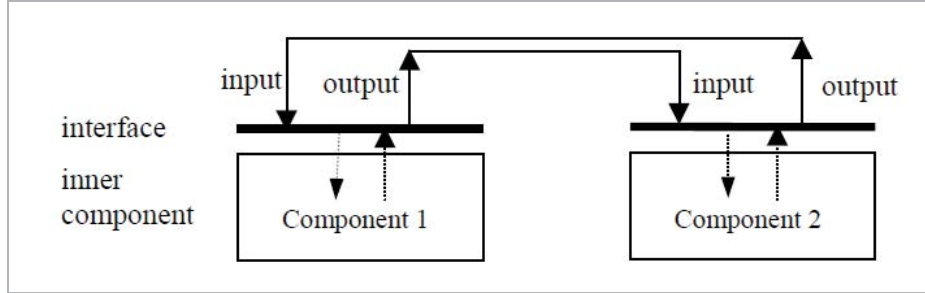
Toplam fonksiyon puanı her bir okuma, yazma, giriş ve çıkışların sayılarının toplamı toplanarak hesaplanır.

Ürün Hattında Arayüz Tabanlı Tasarım

Arayüz tabanlı tasarım, sağlayıcı ve kullanıcı bileşenler arasında arayüzler esas alınarak hazırlanan bir yazılım mimarisidir[9]. Bu tasarım nesne tabanlı yazılım mimarisi sınırları içerisinde kalarak bileşen tabanlı ürün hattı ile uyum içinde çalışması amacı ile yapılmıştır[10]. Ara yüz tabanlı tasarım UML tasarım kavramı ile de uyumludur. Bu metot sayesinde bileşenler Şekil 2'deki gibi belirli arayüzelere uygun olarak ortak bir amacı gerçekleştirmek üzere uyum içinde çalışan çevreden soyutlanmış kutular olarak tasarlanır[11].

Arayüz tabanlı tasarımda, bileşenin kullanıcısı önceden belirlenmiş arayüze uygun olarak bileşene istediği girdileri vererek, bileşen tarafından verilen çıktıları kullanabilmektedir.[9]. Kullanıcı bileşen sınırları içerisindeki işlemlerle

ilgilenmediğinden, bileşenler bir kere test edilerek onaylandıktan sonra sorunsuz olarak herhangi bir yazılıma entegre edilebilmektedir.[11].



Şekil 2 - Bileşen Etkileşimi

Arayüz Tabanlı Tasarım ile COSMIC Yönteminin Birleştirilerek Otomatize Büyüklük Ölçüm Metodu Uygulama

Ürün hatlarında yapılan arayüz tabanlı tasarımlarda arayüzler bileşeni kodlamadan önce belirlenmiş olduğundan yazılımın girdi ve çıktıları büyük oranda belirlidir. COSMIC metodu ile uyumlu olarak düşünüldüğünde bir bileşenin arayüzündeki eleman sayısı bileşenin büyüklüğü ile alakalı olmalıdır. Bu çalışmada bir ürün hattı bileşeninin sağlanmış ve gereksinilmiş arayüzlerini inceleyerek ve COSMIC yöntemi temel alınarak otomatize ve hızlı bir büyüklük tahmini yapılması amaçlanmıştır. Elde edilen sonuçlar COSMIC yöntemiyle manuel olarak bulunan sonuçlar ile karşılaştırılmış ve bir ilişim bulunmaya çalışılmıştır. Bu analizden elde edilecek sonuç hazırlanan otomatize yöntemin ürün hattındaki arayüz tabanlı tasarımlarda büyüklük ölçümünün COSMIC bazında anlamlı sonuç çıkarıp çıkarmadığı açıklığa kavuşacaktır.

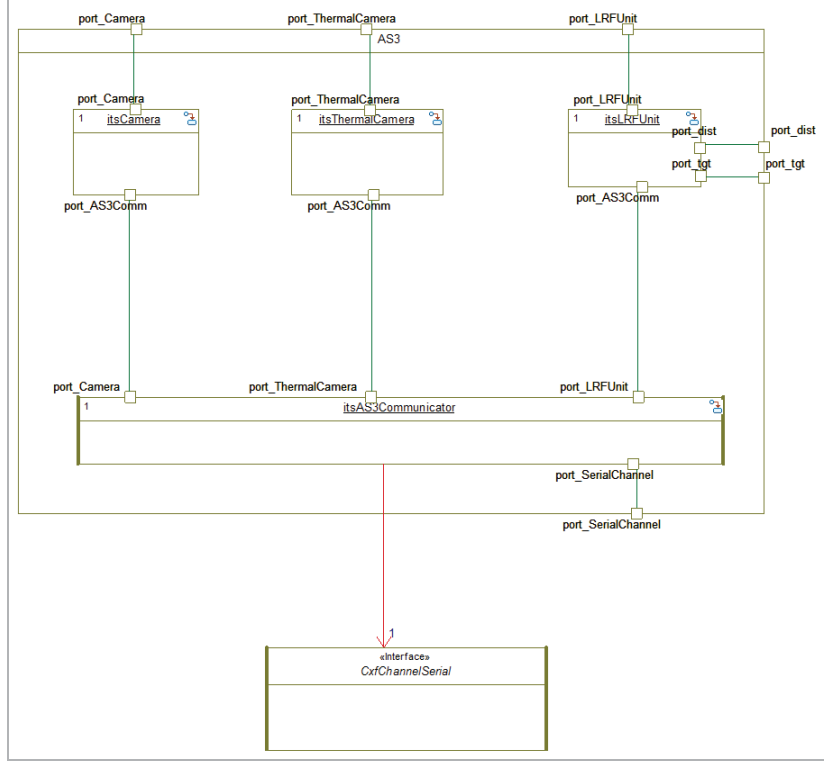
Tablo 1- UML COSMIC İlişkisi

COSMIC Kavramı	UML Karşılığı	UML Elemanı
Yazılım Sınırı	Bileşen Diyagramı	Bileşenin sınırları
Fonksiyonel Kullanıcı	Kullanım Durumu Diyagramı ve Bileşen Diyagramı	Sisteme direkt olarak bağlanan harici bileşenler
Tetikleyen Olay	Bileşen Diyagramı	Harici bileşen tarafından gelen tetikleyici olaylar
Kalıcı Veri Grubu	Bileşen Diyagramı ve Sınıf Diyagramı	Sınıf

Konuk Veri Grubu	Bileşen Diyagramı	Harici bileşenler tarafından gönderilen fonksiyonlar ve bu fonksiyon içerisindeki argümanlar
Süreç	Kullanım Durumu Diyagramı ve Akış Diyagramı	Akış diyagramında gösterilen süreç
Giriş Veri Hareketi	Akış Diyagramı	Harici yazılım bileşeninden gelen olay ve fonksiyonlar (Sağlanmış Ara yüz)
Çıkış Veri Hareketi	Akış Diyagramı	Harici yazılım bileşeninden gelen olay ve fonksiyonlar (Gereksinim Ara yüzü)
Okuma/Yazma Veri Hareketi	Akış Diyagramı	Sınıf içerisinde yer alan kalıcı değişkenler

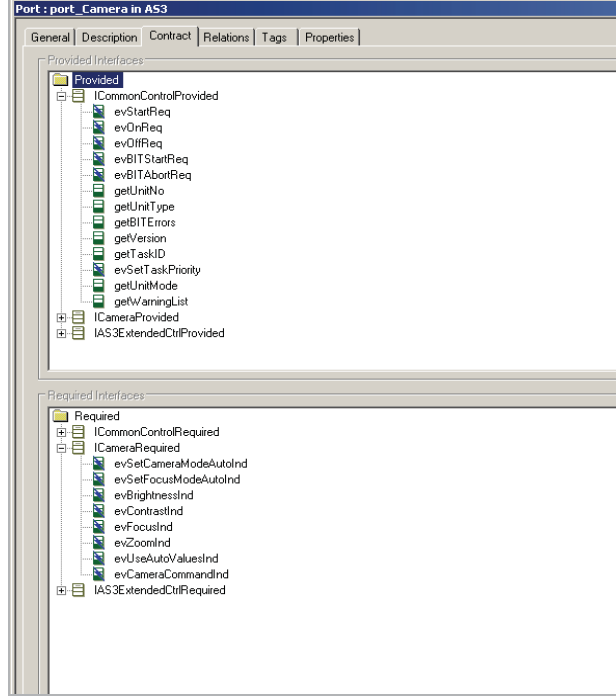
UML COSMIC İlişkisi tablosu göz önüne alındığında UML ve COSMIC metodu birleştirildiğinde arayüz tabanlı tasarım ile hazırlanmış ürün hatlarında bileşenlerin büyüklüğünü hesaplama işlemi otomatize edilebilir. ASELSAN Silah Sistemleri ve Modernizasyonları ekibinin ürün hattında yer alan bileşenler farklı projelerin ihtiyaçlarını karşılamak ve aynı bileşen için tekrar kodlama ve test faaliyeti yürütmek için hazırlanmıştır. Bir projede kullanılacak birim kodu ürün hattında varsa projeden bu ürün hattına referans verilerek kod ortak havuzdan kullanılır.

Bileşen sınırlarını ve bileşen arayüzlerini daha iyi ifade edebilmek için bir bileşenin (Bileşen_7) nesne model diyagramı Şekil 3'te gösterilmiştir. Bileşen_7 bir kamera bileşenidir. Bu kamera paket halinde termal, TV ve lazer içermektedir. Bileşen veri haberleşmesini seri kanal aracılığı ile yapmaktadır. Bileşen içerisinde yapılmış tasarıma göre TV, termal ve lazer işlemlerini yürütebilmesi için 3 adet sınıf ve bu sınıfların seri kanaldan veri gönderebilmesi ve alabilmesi için ortak bir haberleşme sınıfı hazırlanmıştır. Bileşen sınırlarında yer alan portlar bileşeni kullanan sınıflar ile haberleşebilmek için gerekli arayüzleri içermektedir. Bileşenin kullanıcı bileşen sınırları içerisinde yapılan işlerden bağımsız olarak girdilerini bu kanallar üzerinden vererek çıktılarını yine bu kanallardan almaktadır.



Şekil 3 – Bileşen_7'nin Nesne Model Diyagramı

Kamera bileşeni içerisindeki port Thermal portunda yer alan önceden tanımlanmış arayüz ve bu arayüzlerdeki fonksiyon ve olay elemanları Şekil 4'te gösteriliyor. Bu arayüzler daha önce de belirtildiği gibi önceden hazırlanmış arayüzlerdir. ICommonControlProvided arayüzü her bileşen için ortak olan ve her bileşende o bileşenin başlatılması kapatılması hatalarının iletilmesi vb. işlemlerin üzerinden yürütüldüğü arayüzdür. ICameraProvided arayüzü ise kamera bileşenleri için hazırlanan ve yalnızca bir kameranın sağlaması gereken temel fonksiyon ve olayları barındıran arayüzdür. Kamera bileşeninde tanımlanmamış bir fonksiyon sağlanıyorsa genişletilmiş (extended) bir arayüz eklenerek bu özellikler de dahil edilir.



Şekil 4 – Bileşen_7'nin Arayüzleri

3 Durum Çalışması

ASELSAN Silah Sistemleri ve Modernizasyonları ekibinde yazılım büyüklük ölçümü genel olarak geçmiş tecrübelerle elde edilen deneysel verilere dayandırılmıştır. Yeni bir bileşen yazılacağına geçmiş veri göz önüne alınarak bir büyüklük tahmini yapılmaktadır. Ürün hattındaki bileşen arayüzleri iyi tanımlanmış olduğundan yeni bir bileşen yazılacağına kullanılacak arayüzler önceden belirlenmiştir. Dolayısı ile arayüzlerde yer alan girdi ve çıktılar yazılım büyüklüğü ile COSMIC bazında ilişkilendirilebilir elemanlar olmaktadır. Bu çalışmada ihtiyaç duyulan veriyi toplamak için ASELSAN Silah Sistemleri ve Modernizasyonları yazılım ürün hattı kullanılmıştır. Ara yüzde bileşen tiplerine göre 15 adet bileşen seçilmiştir. Bileşenlerin isimleri gizlilik nedeni ile verilmeyerek numaralandırılmıştır.

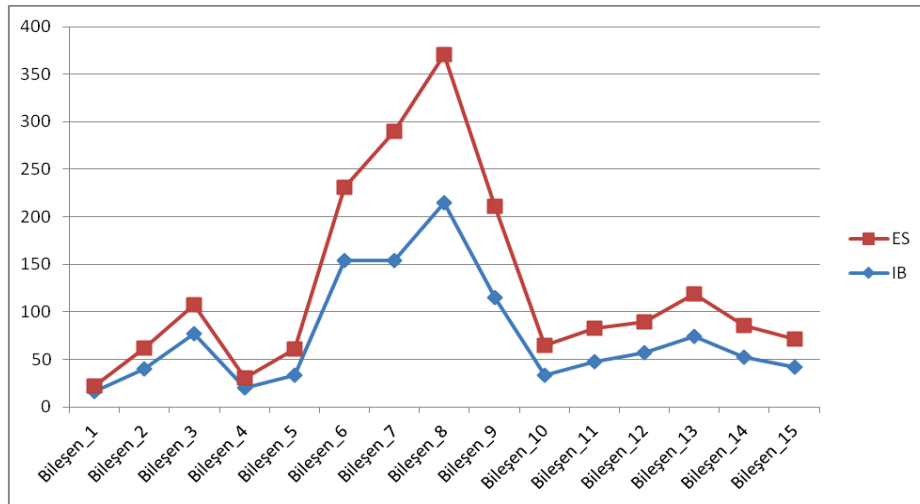
Öncelikle bileşenlerin büyüklüğü el yordamıyla COSMIC işlevsel büyüklük ölçüm metodu ile hesaplanmıştır. Daha sonra ürün hattındaki bileşenlerin arayüzlerindeki eleman sayıları göz önüne alınarak COSMIC ile arayüz elemanlarının sayısı arasındaki ilişkiye bakılmıştır. Tablo 2’de görüldüğü üzere İB kolonu her bir bileşen için el yordamıyla hesaplanan işlevsel büyüklükleri, FS arayüzdeki fonksiyon sayısını OS arayüzdeki olay sayısını ve ES ise arayüzdeki toplam eleman sayısını (FS + OS) göstermektedir. Kullanılan Rhapsody UML aracı için bir eklenti hazırlayarak bu bileşenlerin geçiren (ing. non-behavioral) portlarından giriş ve çıkışlardaki

fonksiyon ve olayların toplamını bulan bir kod geliştirilmiştir. Burada daha önceden bahsedilen COSMIC kavramlarından yazılım sınırı olarak bu bileşenin sınırlarının alındığı ve giriş ve çıkışlar için de harici bileşen ve yazılımlarla haberleşmek için kullanılan olay ve fonksiyonların kullanıldığı görülebilmektedir.

Tablo 2 – Ürün Hattı Bileşen Bilgileri

Bileşen Numarası	İB	FS	OS	ES
Bileşen_1	16	1	5	6
Bileşen_2	40	14	8	22
Bileşen_3	77	12	18	30
Bileşen_4	20	5	5	10
Bileşen_5	33	11	17	28
Bileşen_6	154	50	27	77
Bileşen_7	154	33	103	136
Bileşen_8	215	32	124	156
Bileşen_9	115	30	66	96
Bileşen_10	33	9	23	32
Bileşen_11	48	15	20	35
Bileşen_12	57	17	15	32
Bileşen_13	74	12	33	45
Bileşen_14	52	10	24	34
Bileşen_15	42	14	15	29

Şekil 5’te arayüzdeki toplam eleman sayısına göre değişen işlevsel büyüklük grafiği görülmektedir.



Şekil 5 – İşlevsel Büyüklük ve Eleman Sayısı İlgileşimi

Bileşenler için işlevsel büyüklük ve fonksiyon ve olay sayıları hesaplandıktan sonra bu bileşenler için arayüzdeki elemanların tip ve sayılarından yola çıkarak işlevsel büyüklüğü tahmin etmek amacıyla Çoklu Lineer Yaklaşım (ing. Multiple Linear Regression) analizi yapılmıştır. Çoklu Lineer Yaklaşım metodu bir değişkeni bağımlı diğer değişkenler bağımsız seçilerek yapıldığından bu yaklaşım yöntemi yapılacak işlevsel büyüklük tahmini için uygun bulunmuştur. Çoklu Lineer Yaklaşım metodu ile yapılan analizler sonucunda Tablo-3'te görüldüğü üzere FS ve OS bağımsız değişkenleri İB ile sırasıyla 0,859 ve 0,879 gibi yüksek bir ilişime sahiptirler.

Tablo 3 - Pearson İlgileşimi

		İB	FS	OS
Pearson İlgileşimi	İB	1,000	,859	,879
	FS	,859	1,000	,609
	OS	,879	,609	1,000

Tahmini işlevsel büyüklüğün (TİB) arayüzdeki fonksiyon ve olay sayısı ile arasındaki ilişki, çoklu lineer yaklaşım ile elde edilen Tablo-4'teki katsayılara göre aşağıdaki fonksiyon ile ifade edilmektedir.

$$TİB = 3,588 + 2,313 * FS + 0,921 * OS$$

Tablo-4'te görüldüğü gibi FS ve OS bağımsız değişkenleri istatistiksel olarak anlamlıdır (Sig<0.05).

Tablo 4 - Katsayılar Tablosu

Model		Unstandardized		Standardized		
		Coefficients		Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	3,588	6,960		,516	,616
	FS	2,313	,404	,514	5,725	,000
	OS	,921	,146	,566	6,297	,000

Tablo 5'te görüldüğü gibi işlevsel büyüklüğü ölçmekte kullandığımız tahminin istatistiksel olarak anlamlı olduğu sonucu çıkmaktadır. (F(2,12) = 92.563, p<.001).

Tablo 5 - ANOVA Analizi

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	44716,771	2	22358,385	92,563	,000 ^b
	Residual	2898,562	12	241,547		
Total		47615,333	14			

Çoklu lineer yaklaşım sonucunda elde edilen Tablo 5'teki model özetine göre işlevsel büyüklüğü tahmin etmekte kullanılan fonksiyonda standart tahmin hatasının %15 olduğu görülmektedir. Tablo 5'te görüldüğü gibi "R Square" değeri 0,939'dur. Bu da işlevsel büyüklüğün değişiminin arayüzdeki elemanlar ile %93 oranında açıklanabildiğini göstermektedir.

Tablo 6 - Model Özeti

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	,969 ^a	,939	,929	15,542

ASELSAN Silah Sistemleri ve Modernizasyonları için yukarıda ifade edilen büyüklük tahmini ile elde edilen bileşenlerin tahmini işlevsel büyüklüklerinin hata yüzdeleri Tablo 7'de verilmiştir.

Tablo 7 - İB ve Tahmini İB Karşılaştırması

Bileşen Numarası	İB	Tahmini İB	%Hata
Bileşen_1	16	11	52
Bileşen_2	40	43	8
Bileşen_3	77	48	61
Bileşen_4	20	20	1
Bileşen_5	33	45	26
Bileşen_6	154	144	7
Bileşen_7	154	175	12
Bileşen_8	215	192	12
Bileşen_9	115	134	14
Bileşen_10	33	46	28
Bileşen_11	48	57	15
Bileşen_12	57	57	0
Bileşen_13	74	62	20
Bileşen_14	52	49	7
Bileşen_15	42	50	16

4 Çalışma Kısıtları

Bu çalışma devam eden bir tez çalışmasına girdi sağlama amacı ile gerçekleştirilmiş bir keşif (ing. explorative) çalışmasıdır. Çalışma arayüz tabanlı tasarım ile hazırlanmış bileşen tabanlı ürün hatlarında yapıldığı için sonuçlar başka mimari tasarımlı ürün hatlarında genelleştirilemeyebilir. Çıkan tahmin fonksiyonu çalışmanın yapıldığı ürün hattına özeldir başka bir ürün hattında aynı katsayılar elde edilemeyebilir. Analiz için kullanılan veri kümesi küçüktür, dolayısı ile istatistiksel analizlerden elde edilen sonuçların gücü kısıtlıdır. Ürün hattında ele alınan bileşen sayısı artırılarak işlevsel büyüklük yaklaşımı daha doğru sonuçlar verebilir.

Başka bir çalışma kısıtı da işlevsel büyüklük ölçümlerinin bu çalışmanın ana yazarı tarafından yapılmasıdır. Birden fazla kişi ile yapılması daha güvenli bir işlevsel büyüklük tahmini elde etmeye yardımcı olacaktır.

5 Sonuç ve Gelecek Çalışmalar

COSMIC işlevsel büyüklük ölçüm metodundan elde edilen işlevsel büyüklük arayüz tabanlı tasarımı ile hazırlanmış ürün hatlarındaki bileşenlerin arayüzlerindeki olay ve fonksiyon sayısı ile paralellik göstermektedir. İki bileşenden arayüzünde daha fazla fonksiyon ve olay bulunanın işlevsel büyüklüğünün daha fazla olduğu görülmektedir. Arayüz tabanlı tasarım yapılmış ürün hatlarında yeni bir bileşen hazırlanacağına göre hazırlanacağından bileşenlerin işlevsel büyüklükleri önerilen yöntem ile hızlı ve doğruluk payı kabul edilebilir şekilde tahmin edilebilir. Bu şekilde geçmiş tecrübeye dayandırılan işlevsel büyüklük tahminleri bir metoda oturtularak geçmiş tecrübeye dayalı olmaktan çıkarılır. Geçmiş tecrübe ile yapılan tahminlerde hata payı istatistiki olarak %25 ile %30 civarında bulunmaktadır. Tasarım tabanlı arayüzlerde yer alan elemanlara bakılarak yapılan tahminde ise yapılan hatanın mutlak ortalaması %15 olarak bulunmaktadır. Tahminin doğruluğunun artırılması amacıyla UML'den elde edilebilecek başka girdiler araştırılarak işlevsel büyüklüğün tahminindeki doğruluk payının artırılması planlanmaktadır. UML şemaları ile COSMIC işlevsel büyüklük metodu konseptleri arasında hazırlanan ilişkilendirme detaylandırılarak tahmin yerine ölçümün yapılması planlanmaktadır.

Kaynakça

1. Pressman, R. S. (2001). Software Engineering: A Practitioner's Approach. McGraw Hill. ISBN: 0-07-365578-3
2. Dikel, D., Kane, D., Ornburn, S., Loftus, W. & Wilson, J. (1997). Applying Software Product-Line Architecture.. IEEE Computer, 30, 49-55.
3. Moløkken, K. & Jørgensen, M. (2003). A Review of Surveys on Software Effort Estimation.. ISESE (p./pp. 223-231), : IEEE Computer Society. ISBN: 0-7695-2002-2

4. Verner, J. M. & Tate, G. (1992). A Software Size Model.. *IEEE Trans. Software Eng.*, 18, 265-278.
5. Völter, M. & Groher, I. (2007). Product Line Implementation using Aspect-Oriented and Model-Driven Software Development.. *SPLC* (p./pp. 233-242), : IEEE Computer Society
6. Ziadi, T., H'elou'et, L. & J'ez'equel, J.-M. (2003). Towards a UML Profile for Software Product Lines.. In F. van der Linden (ed.), *PFE* (p./pp. 129-139), : Springer. ISBN: 3-540-21941-2
7. Krajnc, A., Hericko, M., Gerlec, C., Goljat, U. & Polancic, G. (2012). Experimental investigation of the quality and productivity of software factories based development.. *Comput. Sci. Inf. Syst.*, 9, 667-689.
8. COSMIC – Common Software Measurement International Consortium: The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May 2009.
9. Jezequel, J.-M. & Meyer, B. (1997). Design by Contract: The Lessons of Ariane. *IEEE Computer*, 30, 129-130
10. Garion, C. & van der Torre, L. (2005). Design by Contract Deontic Design Language for Multiagent Systems.. In O. Boissier, J. A. Padget, V. Dignum, G. Lindemann, E. T. Matson, S. Ossowski, J. S. Sichman & J. V'azquez-Salceda (eds.), *AAMAS Workshops* (p./pp. 170-182), : Springer. ISBN: 3-540-35173-6.
11. Cheesman, J., Daniels, J. & Szyperski, C. (ed.) (2001). *UML Components - A Simple Process for Specifying Component-Based Software*. Addison-Wesley, M. Klein, and T. W. Malone, "Programming the Global Brain," *Commun. ACM*, vol. 55, no. 5, pp. 41–43, May 2012.