# İşletim İzlerinden Otomatik COSMIC Büyüklük Ölçme – Durum Çalışması –

Muhammet Ali SAĞ, Ayça TARHAN

Hacettepe Üniversitesi
Beytepe, Ankara
muhammetalisag@gmail.com, atarhan@hacettepe.edu.tr

**Özet.** Manuel olarak gerçekleştirilen büyüklük ölçme işlemlerinde ölçme yapanın uzmanlığının sonuçları etkilemesi ve ölçümlerin gerektirdiği maliyet ve zaman gibi problemleri ortadan kaldırılabilmek adına, ölçme işleminin otomatikleştirilmesi önem arzeden bir gereksinimdir. Çalışma kapsamında; COSMIC işlevsel büyüklük yöntemi kuralları çerçevesinde, Java iş uygulamalarının işlevsel süreçlerinin tespiti ve yazılım işlevsel büyüklüğünün otomatik olarak hesaplanması hedeflenmekte ve sonuçları durum çalışması ile değerlendirilmektedir. Önerilen yöntem ile yazılımın işlevsel büyüklüğü, çalışma zamanında yapılan dinamik analiz yöntemiyle elde edilen UML Dizge Diyagramları üzerinden hesaplamaktadır. Dinamik analiz için bağlama yönelik programlamanın Java platformundaki gerçekleştirimi olan AspectJ kullanılmaktadır. Önerilen yöntemi desteklemek için geliştirilen "COSMIC Solver" adlı prototip araçla yöntemin işe yararlılığı durum çalışması üzerinden gösterilmektedir. Çerçevesi prototipe uygun olarak seçilen Java uygulamasının işlevsel büyüklüğü manuel olarak ve önerilen yöntem/araçla ölçülerek sonuçlar karşılaştırılmaktadır.

**Anahtar Kelimeler:** İşlevsel büyüklük, COSMIC yöntemi, AspectJ, UML, Dizge Diyagramı, otomatik ölçme, kaynak kod, dinamik analiz

## 1. Giriş

İşlevsel Büyüklük Ölçme yazılım proje yönetimi süreçlerinde önem arz eden bir konudur. Bilgi sistemlerinin süreç yönetimlerinde spesifikasyonların belirlenmesi, geliştirimi ve gerçekleştirimlerinin etkin şekilde yapılabilmesi, ilgili sistemlerin işlevsel büyüklük bilgilerinin elde edilmesi ve etkin şekilde kullanılmasıyla mümkündür.

ISO 14143-1 [12] standardında yer alan tanıma göre, işlevsel büyüklük ölçümü; bilgi sistemi gereksinimlerinin kullanıcı perspektifinde işlevsel eylemler olarak belirlenmesidir. Albrecht ve ark. [2]'nın yaptıkları çalışma ve önerdikleri yöntem sonrasında, işlevsel büyüklük ölçümü İşlev Puan Analizi ile gerçekleştirilmektedir. İşlev puanı, paydaşlar için kavramsal olarak işlevsel terimleri kullanan bir yazılım metriğidir.

Yazılım geliştirme faaliyetlerinin ilk adımlarından itibaren işlevsel büyüklüğün ölçülebilir olması, ölçme işlemini, artan yazılım boyutları nedeniyle kritik önem

kazanan yazılım proje yönetimi sürecinin en önemli bileşenlerinden biri haline getirmiştir. Diğer yandan geliştirici yeteneği, tasarım tekniği ya da programlama dili değişse dahi işlevsel büyüklük puanının değişmiyor olması, işlev puanı diğer büyüklük ölçümlerinden pozitif ayrıştırmakta ve öne çıkarmaktadır.

Bütün bunlara rağmen, işlevsel büyüklük ölçümü yöntemlerinin henüz çözüme kavuşturulmamış problemleri bulunmaktadır. Bu problemlerden biri, aynı organizasyonda ve aynı üründe olsa dahi farklı uzmanlar tarafından yapılan ölçümlerde elde edilen sonuçların insan faktörü nedeniyle birbirinden farklılık gösterebilmesidir [8]. Diğer yandan ölçme eylemi için gerekli uzmanlık ve eylemin oluşturduğu ekstra maliyet problemi, işlevsel büyüklük ölçümü yöntemlerinin endüstride yaygınlık kazanmasını engellemektedir. Belirtilen problemlere bir çözüm getirebilmek ve öznel yargılamaya neden olan insan faktörünü minimize ederek tutarlı sonuçlar elde edebilmek için, ölçme işleminin otomatikleştirilmesi bir zorunluluk olarak öne çıkmaktadır [2], [3], [4].

Bu bildiride, Java iş uygulamaları için COSMIC işlevsel büyüklük ölçme eylemini otomatikleştiren COSMIC Solver aracı ve aracın kullanıldığı durum çalışmasının detayları anlatılmaktadır. Araç, uygulama kodu üzerinde çalışarak işlevsel süreçlerin tespit edilmesini ve bu işlevsel süreçlere ait veri hareketlerinin oluşturduğu toplam işlev puanın bir uzmana ihtiyaç duymadan hesaplanmasını sağlamaktadır. Bu hesaplamanın yazılım kurumlarında; yazılım modernizasyon projelerinde yeniden geliştirilecek yazılım büyüklüğünü hesaplanmada, yazılım büyüklüğü kütüphaneleri oluşturmada ve üretilen yazılımın işlevsel büyüklüğüne dayalı olarak proje takibi yapmada faydalı olacabileceği değerlendirilmektedir.

Bildirinin izleyen bölümlerinde, makalenin ana konusu olan İşlevsel büyüklük ve uygulamalarından olan COSMIC yönteminin yanısıra kullanılan teknolojilere değinilmektedir. Önerilen yöntem geliştirilirken refererans alınan yakın ilişkili çalışmalar hakkında kısa bir bilgi verilmekte ve yöntem anlatılmaktadır. Son bölümde ise yöntemi doğrulama amacıyla yapılan durum çalışmasına ilişkin geniş bilgiler verilerek sonuçlar değerlendirilmektedir.

# 2 Ön Bilgi

## 2.1 İşlevsel Büyüklük Ölçme

ISO 14143-1 [12] standardına göre, bir bilgi sisteminin işlevsel büyüklüğü işlevsel bilgi sistemi gereksinimlerinin kullanıcı perspektifiyle elde edilmesidir. Albrecht ve ark. yaptıkları öneriyle [2] işlevsel büyüklük, İşlev Puanı Analizi kullanılarak elde edilmektedir. İşlev Puanı, paydaşlar için mantıksal işlevsel terimleri kullanan yazılım büyüklüğü metriğidir. Yazılım büyüklüğünün işlevsel gereksinimlerden hesaplanması, geliştirici yeteneğinden, tasarım tekniğinden ya da programlama dilindeki değişikliklerden bağımsız bir metrik sunmaktadır.

### 2.2 COSMIC İşlevsel Büyüklük

COSMIC yöntemi yazılım geliştirme süreçlerinde yaygın şekilde kullanılan ve diğer yazılım büyüklük ölçüm yöntemlerinin platform bağımlılığı, geliştiriciye bağımlılık vb. zayıf yönlerini taşımayan güncel ve yenilenmiş bir işlevsel büyüklük ölçüm yöntemidir [1].

COSMIC ölçme yönteminin hedefi yazılımların işlevselliğini ölçmektir. İşlevsellik, yazılımın kullanıcılar için gerçekleştirebildiği bilgi işleme eylemlerinin tümüdür. İşlevsel kullanıcı gereksinimleri ("Functional User Requirements – FUR") ise işlevsel kullanıcılar için yazılımın gerçekleştirmesi gereken şeyleri tanımlar. Bu gereksinimler içerisinde teknik ya da kalite gereksinimleri yer almamaktadır.

Ölçme Stratejisi, eşleme ve ölçme evresi olarak adlandırılan üç safhada tamamlanan ölçme sürecinde hedef, temel olarak, İşlevsel kullanıcı gereksinimlerinin ortaya çıkardığı işlevsel süreçlerdeki veri hareketlerinin belirlenerek sayılmasıdır. Veri Hareketi bir işlevsel sürece ait, yalnızca bir veri grubunun hareket etmesine sebep giriş, okuma, yazma ve çıkış olarak tanımlanan eylemlerdir.

### 2.3 Bağlama Yönelik ("Aspect-Oriented") Programlama

Bağlama yönelik programlama (Aspect Oriented Programming) yazılım mühendisliğinde kullanılan bir programlama yaklaşımıdır. Nesneye yönelik programlamanın ele alamadığı ilgiler (concerns) üzerine kurulmuştur. AOP olarak kısaltılan bu yaklaşım Türkçe literatüründe bağlam / cephe / kesit / görünüm yönelimli programlama şeklinde çeşitli kelimelerle ifade edilebilmektedir [16].

### 2.4 AspectJ

Bağlama yönelik programlama kavramının Java platformunda gerçekleştirimi AspectJ'dir. AspectJ, Java platformu için yeni bir modülerlik formu sağlayan uzantıdır. Daha önce belirtilmiş olan, sistemin modülerliğini bozabilen çapraz kesilen kavramların sarmalanmasını sağlamak için tasarlanmıştır.

COSMIC işlevsel büyüklüğün otomatik olarak hesaplanması için önerilen yöntemde AspectJ yoğun olarak kullanılmaktadır. Yöntemin daha iyi kavranması adına AOP ve AspectJ teknolojileri hakkında bilgi sahibi olmak faydalı olacaktır (bkz. 20], [21], [22]).

## 3 İlişkili Çalışmalar

Literatürde bu bildiride anlatılan çalışmaya temel olarak alınan destekleyici çalışmalar mevcuttur. Bunlar UML modelleri kullanarak COSMIC işlevsel büyüklük hesaplamanın nasıl desteklenebileceğine ilişkin önemli bilgiler içermekle birlikte, UML tabanlı çalışmalar kullanarak büyüklük hesaplanmanın nasıl otomatikleştirile-bileceğine dair çeşitli bilgiler ve öneriler de içermektedir. İlişkili çalışmalar

belirlenirken, UML diyagramlarını çeşitli şekillerde manuel ya da otomatik COSMIC büyüklük ölçmede kullanan yazarların çalışmaları seçilmiştir:

- COSMIC ve UML kavramlarınu eşleme çalışma: Bévo ve ark. [6]
- Dizge diyagramlarını kullanarak kullanım senaryolarının taneciklik problemini adresleyen çalışma: Jenner ve ark. [7]
- İşlevsel büyüklüğün ölçülmesinde otomasyon için UML kavramlarının açılmasına dayanan çalışma: Vilus ve ark. [18]
- Kullanım durumu modeliyle inşa edilmiş ölçme çalışması: Habela ve ark. [17]
- Rational Unified Process (RUP) yöntemi kullanılarak otomatik olarak COSMIC tabanlı işlevsel büyüklük hesaplaması: Azzouz ve ark. [11]
- COSMIC büyüklük ölçümünü desteklemek için UML kullanılabilirliğini inceleyen çalışma: Lavazza ve ark. [10]
- UML in İşlevsel Büyüklük ölçümüne nasıl destek olabileceğine ilişkin bir gereksinimler uzayı çalışması: Van den Berg ve ark. [18]
- UML kullanım durumları ve dizge diyagramları üzerinden elle ölçmenin kolaylaştırılmasına ilişkin bir çalışma: Levesque ve ark. [13]
- Dizge diyagramı elementlerinin COSMIC artefaktlarıyla eşleştirilmesini temel alan çalışmalar: Fehlmann ve ark. [9]

# 4 Yöntem

COSMIC işlevsel büyüklük hesaplanması karşılaştırma yapılabilmesi adına çok önemli proje girdileri sunabilmektedir. Fakat daha önce geliştirilen yazılımların işlevsel büyüklük bilgisine sahip olunmaması, dokümantasyon eksikliği veya geliştirilen yazılıma hakim kimsenin kalmaması gibi durumlarda yazılımın işlevsel büyüklüğünün nasıl hesaplanabileceği konusu büyük bir soru işareti oluşturmaktadır. Manuel ölçümlerin alacağı süre ve oluşturacağı maliyet, geçmişe yönelik böyle bir çalışmanın yapılmasına engel teşkil edebilmektedir.

Çalışmanın temel motivasyonunu teşkil eden "Bir uygulamanın büyüklüğünü koda müdahale etmeden ve uygulamayı kapalı kutu gibi ele alarak nasıl ölçebiliriz?" sorusu bizi daha önce belirtilen durumları göz önünde bulundurarak değinilen problemlere çözüm bulabilmek adına iki farklı soruya yönlendirmektedir:

1) Çalışan bir uygulamadan UML dizge diyagramları elde edebilir mi?
2) UML dizge diyagramları kullanılarak COSMIC puan otomatik ölçülebilir mi?

Sorulan sorular çerçevesinde yapılan literatür taramasıyla Jenner [7] ve Lévesque [13] adlı araştırmacıların UML dizge diyagramları ve COSMIC büyüklük kavramlarını eşleştirme çalışmalarıyla dizge diyagramlarının COSMIC işlevsel büyüklüğün ölçülmesi için uygun yeterlilikte olduğunu ispatlamış olduklarını görülmüştür. Lévesque ve ark. [13] 'nın veri hareketlerinin UML dizge diyagramlarının mesajlarının eşleniği olarak ele alınabileceğini ve aynı diyagramda veri işleme hareketlerinin çeşitli şekillerde hata mesajları gibi ele alınabileceğini göstermesi çalışmanın yönünü belirlemiştir.

İlk sorumuz olan "çalışan bir uygulamadan UML dizge diyagramlarını elde edebilir miyiz?" sorusunun ortaya çıkardığı problem diğer kısıtlar da göz önünde bulundurarak bağlama yönelik programlamanın sunduğu olanaklar ile çözülmüştür.

Bağlama yönelik programlama ile çalışma zamanında dizge diyagramları elde edilmektedir.

Ayrıntıları daha önceki çalışmamızda yer alan önerilen yöntem [19] özetle dört adımdan oluşmaktadır.

- *Kesme Noktalarının Hazırlanması* – Java platform ve AspectJ sözdizimi kullanarak birleşme noktalarının tanımlanması. Daha sonra veri hareketleri ve işleme eylemlerini yakalayabilecek kesme noktalarının tanımlanması. (Java, AOP ve COSMIC ölçüm yöntemi konusunda uzman tarafından)
- *Dinamik Analizin Yapılması* – Yükleme zamanında yapılan dokuma prosedürü ile hedef uygulamanın birinci adımda tanımlanmış tavsiye yordamlarıyla normal bir kullanıcı olarak kullanılması. (Normal kullanıcı/ölçüm yapan kişi tarafından)
- *Dizgelerin Metin Formunda Elde Edilmesi* – Ölçülmesi hedeflenen her bir işlevsel sürecin tetiklenmesi sonucu etiketlenmiş bir akış stilinde dizge diyagramının metin halinde elde edilmesi. (Normal kullanıcı/ölçüm yapan kişi tarafından)
- *COSMIC Kurallarının Uygulanması ve Büyüklüğün Ölçülmesi* – Metin formunda oluşturulan dizge diyagramlarındaki işlevsel süreçlerin tespit edilerek COSMIC kuralları çerçevesinde sayma işlemlerinin gerçekleştirilmesi (Normal kullanıcı/ölçüm yapan kişi tarafından)

# 5 Durum Çalışması

Önerilen "Kod Üzerinden Otomatik Olarak COSMIC Büyüklük Hesaplama Yönteminin" kendi geliştirdiğimiz küçük bir prototip uygulama üzerinde çalışabilirliğini önceki makalemizde [19] gösterdikten sonra, yöntemin üçüncü bir kişi/organizasyon tarafından geliştirilmiş daha geniş kapsamlı bir yazılım üzerinde test edilebileceği bir durum çalışmasının yapılmasına karar verilmiştir. Bu durum çalışmasında cevap aranan araştırma soruları (AS) şunlardır:

AS-1. Önerilen yöntem işlevsel büyüklüğük hesaplamasında etkili midir?

AS-2. Önerilen yöntem zaman maliyeti açısından etkin midir?

## 5.1 Durum Çalışması Hazırlıkları

Durum çalışması için hazırlanan sorulara cevap bulabilmek için bir ön çalışma yapılması gerekmektedir. Bu ön çalışma önerilen yöntemin uygulanabileceği uygulamaların araştırılması ve kriterlere uygunluğunun test edilerek onaylanması süreçlerini içermektedir. Bu çalışmalar neticesinde üçüncü parti tarafından geliştirilmiş bir Java iş uygulamasının belirlenen sorular çerçevesinde önerilen yöntemle ölçülmesi işlemi gerçekleştirilmiştir.

Çalışmanın hedeflerinden gelen kriterlerin yanısıra, önerilen yöntemin prototip olması nedeniyle ortaya çıkan kısıtların belirlenmesi ve bu kısıtlar çerçevesinde arama yapılması gerekmektektedir. Ölçümü yapılacak uygulamanın belirlenmesinde göz önünde bulundurulan kısıtlar şunlardır:

- Uygulamanın Java programlama diliyle geliştirilmiş olması gereklidir.

- Uygulamanın Java SE 1.5 ve üzeri sürümle geliştirilmiş olması gereklidir.
- Uygulama ara yüzünün Java Swing ile geliştirilmiş olması gereklidir.
- Uygulamanın iki ya da üç katmanlı mimari ile geliştirilmiş olması gereklidir.
- Uygulamanın veritabanı işlemleri ağırlıklı bir yapıda olması gereklidir.
- Uygulama, kullanılması durumunda JAX-RPC, JPA, JDBC standartlarına uygun geliştirilmiş kütüphaneler kullanıyor olmalıdır.
- COSMIC Manuel'de [1] yeralan "COSMIC Yazılım Bağlamı Model İlkeleri" ve "Jenerik Yazılım Modeli İlkeleri" başlığı altında yeralan şartları sağlayan bir uygulama olması gerekmektedir.

Uygulamanın üçüncü kişi ve kuruluşlar tarafından geliştirilmiş olması önerdiğimiz yöntemin uygulamaya özgü sonuçlar üretip üretmediğini test etmek açısından önemli ve sonuçların güvenilirliği için gereklidir. Bu yüzden belirtilen kriterlere uygun bir tarama gerçekleştirilmiştir.

Amacın gerçekleştirilmesi ve durum çalışması için hazırlanmış sorulara cevaplar için açık kaynak yazılımlara başvurulmaktadır. Açık kaynak yazılımların sunulduğu, *GitHub*, *SourceForge*, *GoogleCode* gibi yazılım depoları taranarak aday yazılımlar tespit edilmiştir. Daha sonra aday yazılımlar kriterler doğrultusunda seçme işlemine tabi tutularak durum çalışmasının uygulanacağı yazılım belirlenmiştir.

Ölçüm yapılacak yazılım, veritabanı ağırlıklı işlemlerin gerçekleştirildiği Java programlama diliyle geliştirilmiş, Swing arayüz kütüphanelerini kullanan bir uygulamadır.

### 5.2 Durum Çalışması Gerçekleştirimi ve Sonuçları

Aday uygulamalar içinden *SourceForge* kaynaklı "Nyagua Aquarium Application", durum çalışması için hedef uygulama olarak seçilmiştir. Uygulama deposu üzerinde uygulamanın sadece kaynak kodu yer almakta olup, uygulamaya ait dokümanlara ulaşılamamaktadır. Bu durum genel bir problem olup otomatikleştirme yöntemi önerisinin de faydalarından birini oluşturmaktadır. Dolayısıyla gerçek bir vaka üzerinden sadece kaynak kodu kullanarak büyüklüğün hesaplanması hedefini sağlayabilen bir durum çalışması olacaktır.

#### 5.2.1 Birinci Adım: Bilgi Edinme

Eylem planının birinci adımında yer alan maddeler gereğince uygulama geliştiricisi sitesinden ve SourceForge uygulama deposundan uygulama ile ilgili bilgiler elde edilmiştir. Bu bilgiler Tablo I ve II'de gösterilmektedir.

**Tablo I.** Durum Çalışması İçin Belirlenen Uygulamaya Ait Genel Bilgiler

| | |
|---|---|
| **Uygulama Adı:** | Nyagua Aquarium Management Application |
| **Uygulama Tanımı:** | Akvaryum sahiplerinin / yöneticilerin işlemlerini kolaylaştıran; bir akvaryumda yer alan, omurgasızlar balık, bitki gibi canlıların takibi ve çeşitli ölçümlerin takip imkanı sunan bir yönetim destek uygulamasıdır. |
| **Platform:** | GNU/Linux olarak geliştirilen uygulama Java uygulama geliştirme dilinin sahip olduğu özellikler nedeniyle hemen her işletim sisteminde çalışabilecek şekilde adaptasyona sahiptir. |
| **Lisans:** | Uygulama GNU GPL v.2 lisansına sahiptir. Dolayısıyla ücretsiz dağıtıma sahiptir. |

**Tablo II.** Durum Çalışması İçin Belirlenen Uygulamaya Ait Teknik Bilgiler

| | |
|---|---|
| **Java Sürümü:** | Java SE 1.7 |
| **GUI:** | Java Swing |
| **Veritabanı:** | SQLITE |
| **JDBC / Persistence:** | Native JDBC |
| **JDBC URL:** | jdbc:sqlite://localhost/<file> |
| **Geliştirme Platformu:** | NetBeans |
| **Kullanılan Kütüphaneler:** | sqlite-jdbc-3.7; Jcalendar – 1.4; JDK 1.7 |

**Uygulama kullanım bilgileri;**
Uygulama kaynak kod olarak sunumu NetBeans projesi şeklindedir. Dolayısıyla ücretsiz olarak indirilip kullanılabilecek olan uygulama, NetBeans geliştirme ortamına "import" edilerek derlenebilir. Uygulamayı derlemeden önce sistemde Java JDK 1.5+ geliştirme kütüphanesi yüklü olmalıdır. Ayrıca, uygulamanın SourceForge uygulama deposundaki tanıtım sayfasında yeralan indirmeler bölümünde halihazırda derlenmiş hali de sunulmaktadır. Basit bir kurulum sihirbazı eşliğinde sisteminize yüklenen uygulama, Java Sanal Makinesi ile koşturulmaktadır.

**Uygulamanın kriterleri sağlayıp sağlamadığının kontrolü;**
Uygulamanın belirtilen kriterler açısından değerlendirilebilmesi için kaynak koduna veya uygulama dokümanlarına ihtiyaç bulunmaktadır. Elimizde uygulamaya ait dokümanlar bulunmadığından kriter kontrolü kaynak kod üzerinden gerçekleştirilmiştir. Buna göre uygulamanın Java programlama diliyle gerçekleştirilmiş olması, Swing arayüzünü kullanıyor olması, veritabanı ağırlıklı işlevsel gereksinimler sunması, iki katmanlı bir mimari sunması vb. gibi kriterleri sağladığı tespit edilmiştir.

### 5.2.2 İkinci Adım: Manuel Hesaplama

Uygulamanın, kaynak kod kullanılarak manuel şekilde işlevsel büyüklüğü tespit edilmiştir. İşlevsel büyüklüğün yanı sıra işlevsel süreçlerin belirlenmesi ve kriterler kapsamında toplam işlevsel büyüklüğe eklenip eklenmeyeceğine karar verilmiştir. Yapılan çalışmada elde edilen bilgiler Tablo III' de yer almaktadır.

**Tablo III** Durum Çalışması İçin Belirlenen Uygulamaya ait Manuel İşlevsel Büyüklük Ölçüm Bilgileri

| | |
|---|---|
| Uygulamada yer alan işlevsel kullanıcı gereksinimi sayısı: | 45 |
| Kriterler kapsamında değerlendirmeye alınan ve manuel olarak büyüklükleri tespit edilen işlevsel kullanıcı gereksinimi sayısı: | 39 |
| Değerlendirmeye alınmış işlevsel kullanıcı gereksinimlerine karşılık gelen işlevsel süreçlerin büyüklüğü: | 710 |
| Toplam İşlevsel büyüklüğün hesaplanmasında harcanan süre: | **~6 saat** |

* Değerlendirmeye alınmayan 6 işlevsel kullanıcı gereksinimi (Bkz: Tablo VI), daha önce belirtilen kriterleri (Bkz: Sf. 12) sağlamadığı için değerlendirmeye alınmamıştır. Değerlendirmeye alınabilmesi için prototipte henüz devreye alınmamış dosya operasyonları modülünün tamamlanmış olması gerekmektedir.

### 5.2.3 Üçüncü Adım: Otomatik Hesaplama

Uygulamanın otomatik ölçülmesi işleminde, ilk önce hedef uygulama kütüpha-neleri ile birlikte Eclipse 4.5 AJDT geliştirme yazılımına import edildi. Yöntem çerçevesinde geliştirilen kesme noktaları ("Pointcut") paketi uygulamaya dahil edildi ve proje AspectJ uygulamasına dönüştürüldü. Uygulama çalıştırma yöntemi olarak yükleme zamanı dokuma ("LTW-Load-time Weaving") modu seçildi ve kesme noktaları ("Pointcuts") dizini çalıştırma konfigürasyonuna eklendi. Ardından uygulama çalıştırılarak ölçülmesi hedeflenen tüm süreçler bir işlevsel kullanıcı modunda koşturuldu. "AspectJ LTW" modunda çalıştırılan uygulama kesme noktaları vesilesiyle uygulamanın metin tabanlı UML dizge diyagramını üretildi. Son olarak prototip olarak geliştirilen CosmicSolver uygulaması çalıştırılarak üretilen çıktının işlenmesi ve hesaplama işlemlerinin gerçekleştirilmesi sağlandı. Geliştirilen CosmicSolver ile hedef uygulama için gerçekleştirilen hesaplamaya ait sonuçlar Tablo IV'de yer almaktadır.

**Tablo IV** Durum Çalışması İçin Belirlenen Uygulamanın *CosmicSolver* ile Elde Edilen İşlevsel Büyüklük Ölçüm Bilgileri

| | |
|---|---|
| Uygulamanın yakaladığı aday işlevsel süreç sayısı: | 41 |
| Asil işlevsel süreç sayısı: | 40 |
| Değerlendirmeye alınmış işlevsel kullanıcı gereksinimlerine karşılık gelen işlevsel süreçlerin büyüklüğü: | 719 |
| Toplam İşlevsel büyüklüğün hesaplanmasında harcanan süre: | 30 dk.* |

** Normal şartlar altında (bu süreye uygulamanın çalıştırılmasına ilişkin hazırlıklar dahildir.)*

### 5.2.4 Dördüncü Adım: Değerlendirme

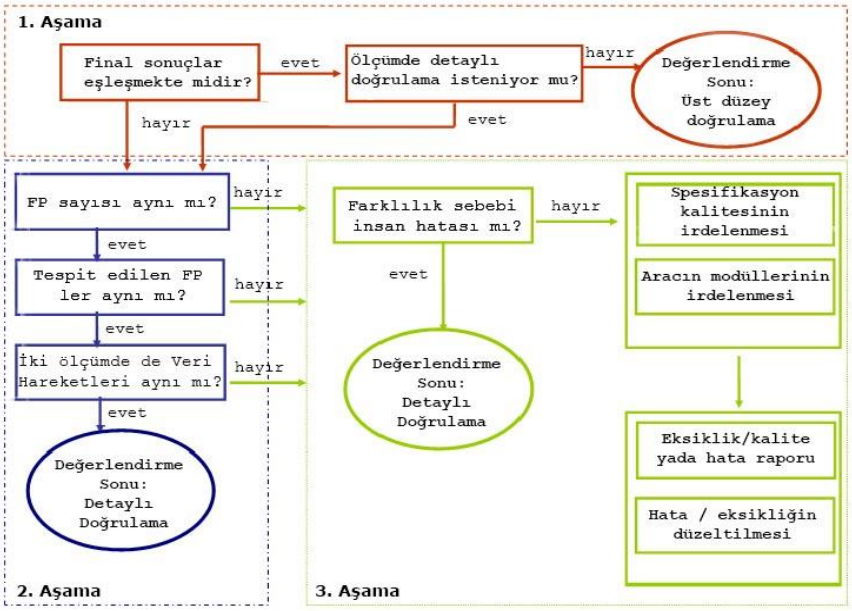
Elde edilen sonuçların detaylı olarak karşılaştırılabilmesi ve sonuçların doğruluğunun teyidi için Soubra ve ark. [15] makalesinde önerilen yöntem izlenmiştir. Soubra ve ark. makalesinde otomatik işlevsel büyüklüğün hesaplanmasında elde edilen sonuçların doğrulaması için üç aşamalı bir yöntem önermektedir. Bu yöntemi kısaca bir akış şeklinde özetlemek gerekirse Şekil 1'de belirtilen adımlar izlenerek sonuçlar karşılaştırılmalıdır.

**Birinci Aşama: Üst Düzey Doğrulama**

Önerilen yöntemde ilk olarak toplam büyüklüğün otomatik ve manuel ölçümlerden elde edilen sonuçlarını dikkate alarak bir yönlendirme yapılmaktadır. Toplam sonuç değerler birbirine eşit ise üst düzey doğrulamanın sağlanabildiği düşünülmekte ve işlem sonlandırılmaktadır.

Geliştirilen prototipin durum çalışması ölçümünde elde ettiği sonuçlar, değerlendirmeye alınan işlevsel kullanıcı gereksinimlerinin uzman tarafından elle yapılan ölçme işlemi sonuçlarıyla paralellik göstermektedir. Bu durum, üst düzey doğrulama aşamasında, değerlendirmede göz önünde bulundurulan toplam büyüklük değerinin önerilen yöntem kullanılarak elde edilen sonuçlara çok yakın olması hasebiyle başarılı olarak nitelendirilebilir.

| **Elle Yapılan Ölçüm** | **Otomatik Olarak Yapılan Ölçüm** |
|---|---|
| Toplam COSMIC Puan: 710 | Toplam COSMIC Puan: 719 |




**Şekil 1** Doğrulamaya ilişkin aşamalar ve adımlar [15]

### İkinci Aşama: Detaylı Doğrulama

Birinci aşama temel düzey olup genel anlamda bir bilgi verse de daha detaylı bilgiler uygulamanın değerlendirilmesinde etkin rol oynamaktadır. Soubra ve ark. bu aşamada sırasıyla işlevsel süreçlerin sayısının, tespit edilen işlevsel süreçlerin aynı olup olmamasının ve veri hareketlerinin sayılarının ayrı ayrı ele alınması gerektiğini belirtmektedir. Bu değerlendirme sonucunda önerilen yöntemin ve uygulamanın sağlamlık ve doğruluğu güvence edilmiş olacaktır.

Değerlendirmenin sağlıklı yapılabilmesi ve manuel ölçme işlemlerinde kolaylık sağlaması açısından değerlendirmeye alınan kullanım durumları ("use cases") uzman tarafından belirlenmiştir. Buna göre uygulamada yeralan ve ölçümü gerçekleştirilecek kullanım durumları listesi Tablo V'de belirtilmektedir. Uygulamada yer alan fakat prototipin kısıtları nedeniyle ele alınmayan işlevsel süreçlerin listesi ve ele alınmama sebepleri Tablo VI'da verilmiştir.

Büyüklüğü otomatik ve manüel olarak ölçülen "Nyagua Aquarium Management" yazılımının, işlevsel kullanıcı gereksinimleri bazında farklılık gösteren COSMIC büyüklüğü, Tablo VII'de gösterilmektedir. Sonuçlar incelendiğinde manuel ve geliştirilen araçla otomatik olarak elde edilen işlevsel kullanıcı gereksinimleri sayısında ve bazı işlevsel kullanıcı gereksinimlerinin büyüklük puanlarında farklar olduğu gözlemlenmiştir.

İşlevsel kullanıcı gereksinimleri sayısı elle yapılan ölçüme göre bir fazladır. Toplam puanın farkının bir bölümü buradan gelmektedir. Diğer farklılık ise veri hareketlerindedir. İki işlevsel süreçte meydana gelen farklılık "çıktı" veri hareketi tipindedir. Karşılaştırılan değerler iki boyuttadır; ilki işlevsel süreç sayısı, diğeri ise toplam işlevsel büyüklüktür. Elle yapılan ölçüme göre toplam işlevsel büyüklük

doğruluk oranı yaklaşık %98'dir. İşlevsel süreç sayısı açısından yapılan karşılaştırmaya göre ise bu oran %97'dir.

**Tablo V** Değerlendirmeye alınan Nygua Aquarium Management Uygulaması Kullanım Durumları

| Kullanım Durumları | Object-of-Interest (İlgi Nesnesi) |
|---|---|
| **Ekle, Sil, Güncelle, Getir** | Aquarium |
| **Ekle, Sil, Güncelle, Getir** | Maintenance |
| **Ekle, Sil, Güncelle, Getir** | FishBase |
| **Ekle, Sil, Güncelle, Getir** | InvertebrateBase |
| **Ekle, Sil, Güncelle, Getir** | PlantBase |
| **Ekle, Sil, Güncelle, Getir** | Fish |
| **Ekle, Sil, Güncelle, Getir** | Plant |
| **Ekle, Sil, Güncelle, Getir** | Invertebrate |
| **Ekle, Sil, Güncelle, Getir** | Expense |
| **Ekle, Sil, Güncelle, Getir** | Measurement |
| **Ekle, Sil, Güncelle, Getir** | Device |
| **Ekle, Sil, Güncelle, Getir, İptal, Sonlandır** | Schedule |

**Tablo VI** Değerlendirmeye alınmayan Nygua Aquarium Management Uygulaması Kullanım Durumları

| Kullanıcı Gereksinimi | Değerlendirmeme Sebebi |
|---|---|
| **Export Aquarium Data** | Prototip uygulama dosya okuma yazma işlemlerini ele alacak şekilde yapılandırılmadığı için ele alınmamıştır. |
| **Import Aquarium Data** | |
| **Backup Aquarium Data** | |
| **Convert metrics** | Prototipte ele alınmayan Swing arayüz bileşenleri nedeniyle dahil edilmemiştir. |
| **Nutrient calculator** | |
| **Calculators** | |

**Tablo VII** Nygua Aquarium Management yazılımının otomatik ve manuel ölçmede farklılık gösteren işlevsel kullanıcı gereksinimleri ve büyüklük puanları

| İşlevsel Süreçler | Otomatik Büyüklük | Manuel Büyüklük | Fark |
|---|---|---|---|
| getAquariumDataFromTable | 5 | 0 | 5 |
| invertebratebaseSave | 11 | 9 | 2 |
| invertebratebaseDelete | 11 | 9 | 2 |
| **Toplam:** | **719** | **710** | **9** |

### Üçüncü Aşama: Düzeltme & Geliştirme

Aşamalar arasında bağıntı döngüsel olup aşamalarda verilen cevaplara göre eylemler dallanmaktadır. Üçüncü aşama olarak tanımlanan fazda düzeltmeler, iyileştirmeler ve eklentiler ile doğruluk oranının artırılması hedeflenmektedir. Programlama dillerinin esnek yapısı ve sürekli gelişmesi nedeniyle olası kapsam dışı eylemlerin tespiti ve gerekli düzeltmeler ve eklemeler yapılması bir zorunluluk olarak ortaya çıkmaktadır.

Durum çalışmasında hedef uygulama, prototip kısıtlarına uygun olarak seçildiği için prototipte geliştirme ve genişletme işlemlerine ihtiyaç duyulmamakla birlikte, geliştirici kaynaklı birkaç hatanın düzeltilmesi gereği ortaya çıkmıştır. Prototipte gerçekleştirilen hata düzeltmeleri sonrasında tekrarlanan ölçme işlemleriyle hedefe ulaşılmıştır. Her yeni uygulama ve geliştirme sayesinde gelişme yaşayan prototipin sağlam bir temele oturması, hataların azalması ve genişletme maliyetinin sıfıra yakınsaması mümkün olacaktır.

## 6 Sonuçlar

Durum çalışması ile COSMIC büyüklüğün otomatik olarak kaynak kod üzerinden hesaplanabilmesi için önerilen yöntemin işe yarar olduğu tespit edilmiştir. Yüksek bir oranda (%97) doğru hesaplama yapılabilmesi, önerilen yöntemin işlevsel büyüklüğük hesaplamada etkili olduğunu göstermektedir (AS-1).

Diğer yandan sonuçları ve farklılıkları daha iyi anlayabilmek adına bir kritik yapılacak olursa; farklılıkların kaynağının iki noktada ortaya çıktığı görülebilir: Bunlar; 1) Olası işlevsel süreç sayısı farklılığı ve 2) Veri hareketleri sayısı farklılığıdır. Veri hareketi farklılığı ise 2.a) İşlevsel süreç için Girdi (Entry) sayısı farklılığı ve 2.b) İşlevsel süreç için Çıktı (Exit) sayısı farklılığı olarak ayrılmaktadır. Önerilen yöntemde ortaya çıkan işlevsel süreçlerin sayısının farklılığı önerilen yöntemin ölçme biçiminden (eylem tabanlı) ve kullanıcının arayüzden erişim yaparak işlevsel süreçleri tetikleyici eylemlerde bulunması, dolayısıyla kontrol grupları, eylemleri vb. hareketlerin de aday süreçler olarak yapısal metin formundaki dizge diyagramlarına çıktı olarak yansıması gibi durumlardan kaynaklanmaktadır. Veri hareketlerinin sayısında ortaya çıkan farklılık ise girdi ve çıktı tipindeki hareketlerde görülebilmektedir. Girdi ve çıktıların tespiti otomatik ölçme işleminde en zor adımı oluşturmaktadır. Girdi ve çıktıların çok farklı şekillerde meydana gelebilecek olması işi zorlaştıran nedendir. Önerilen yöntemde girdi ve çıktıların kontrol altında tutulabilmesi için prototipin kapsamı belirlenmiş ve çalışma için Java Swing arayüzü ile limit koyularak çalışma yapılmıştır. Yine de durum çalışmasında bazı işlevsel süreçlerde (9 ve 10 nolu süreçler için) farklı girdi/çıktı sayısına rastlanılmıştır. Bu durumun sebepleri araştırıldığında karşımıza iki neden ortaya çıkmaktadır. Birincisi, prototipteki geliştirici kaynaklı hatalar (“bug”), diğeri ise Swing grafiksel arayüz bileşenlerinin tümünün prototip kapsamında değerlendirmeye alınmamış olmasıdır.

Ölçme zamanı yönünden değerlendirilecek olursa, önerilen yöntemin yaklaşık 1/10 oranında bir kazanç sağladığı ve oldukça etkin olduğu görülmüştür (AS-2). Yazılım olgunluğa eriştikçe ve kapsamı genişledikçe Soubra ve ark.’ın [15] önerdiği doğrulama sürecinde ihtiyaç duyulan döngülerin oluşturacağı zaman kayıpları önlenecek ve ölçme süresi tüm ölçümler için standart hale gelecektir.

Sonuç olarak, prototipin doğası nedeniyle var olan eksiklikler ve geliştirici kaynaklı hataları ayrı tutarsak elde edilen sonuçlar oldukça yüksek doğruluk oranına sahip olmakla birlikte ölçme için ayrılan zaman önerilen yöntemde minimumdur. Yapılan durum çalışması yöntemin başarısını teyit etmekte ve hedeflenen sonuca ulaşıldığını göstermektedir.

## Kaynaklar


1. 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), 2009
2. A. J. Albrecht: Function point analysis, Encyclopedia of Software Engineering, 1. John Wiley & Sons, 1994
3. Akca, Ahmet Ata, and Ayca Tarhan. "Run-Time Measurement of COSMIC Functional Size for Java Business Applications: Is It Worth the Cost?" (IWSM-MENSURA), 2013 Joint Conference of the 23rd Int. Workshop on. IEEE, 2013.
4. Robiolo, G., "How Simple is It to Measure Software Size and Complexity for an IT Practitioner?," Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on , vol., no., pp.40,48, 22-23 doi: 10.1109/ESEM.2011.12, 2011
5. S. Kusumoto et al., "Function point measurement from Java programs," proceedings of ICSE, 2002
6. Bévo, V., Lévesque, G., Abran, A.: Application de la méthode FFP à partir d'une spécification selon la notation UML In: Proc. 9th Int. Workshop Soft. Measurement. Lac Supérieur,Canada, 230-242, 1999
7. Jenner, M.S.: COSMIC-FFP 2.0 and UML: Estimation of the Size of a System Specified in UML - Problems of Granularity. In: Proc. 4th Eur. Conf. Soft. Measurement and ICT Control. Heidelberg, 173-184, 2001
8. B. A. Kitchenham: The problem with function points", IEEE Software, 14(2):, pp. 29-31, 1997
9. Fehlman, T. M., and Eberhard Kranich. "COSMIC Functional Sizing based on UML Sequence Diagrams." MetriKon, Kaiserslautern, 2011
10. Luigi Lavazza, and Vieri Del Bianco. IWSM/Mensura, volume 5891 of Lecture Notes in Computer Science, page 101-121. Springer, 2009
11. Azzouz, S. and A. Abran, "A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC FFP," Software Measurement Euro. Forum Rome, Italy, 2004.
12. ISO. Information technology – Software measurement – Functional size measurement. Part 1: Definition of concepts. International Standard ISO/IEC 14143-1, 2007
13. Levesque, G., Bevo, V., & Cao, D. T. Estimating software size with UML models. Proceedings of the 2008 C3S2E Conference on - C3S2E '08, 7. doi:10.1145/1370256.1370268, 2008
14. A. J. Albrecht, "Measuring Application Development Productivity," Proceedings of the Joint SHARE, GUIDE, and IBM Application Dev. Symposium, Monterey, IBM Corporation, pp. 83–92, 1979
15. H.Soubra, A.Abrain, Verifying the Accuracy of Automation Tools for the measurement of software with COSMIC – ISO 19761, 2014
16. Ramnivas Laddad. AspectJ in Action: Enterprise AOP with Spring Applications (2nd ed.). Manning Publications Co., Greenwich, CT, USA., 2009
17. Habela P., Glowacki E., Serafinski T., Adapting Use Marry Model for COSMIC-FFP based Measurement, in the 15th International Workshop on Software Measurement,Montreal, Canada,Shaker-Verlag, 2005
18. Vilus L, Levesque G., 2004, Une méthode efficace pour l'extraction des instances de concepts dans une spécification UML aux fins de mesure de la taille fonc-tionnelle de logiciels. ICSSEA, 2004
19. Muhammet Ali Sag, Ayca Tarhan, "Measuring COSMIC Software Size from Functional Execution Traces of Java Business Applications", IWSM-MENSURA, 2014, 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA) 2014, pp. 272-281, doi:10.1109/IWSM.Mensura.2014.29
20. Ramnivas Laddad. AspectJ in Action: Enterprise AOP with Spring Applications (2nd ed.). Manning Publications Co., Greenwich, CT, USA., 2009
21. Wang, Yi; Zhao, Jianjun (July 2007). "Specifying Pointcuts in AspectJ" (PDF). Proceedings of the 21st Annual International Computer Software and Applications Conference 2: 5–10. doi:10.1109/ COMPSAC.2007.196. ISBN 0-7695-2870-8, 2010
22. Dean Wampler, Use Cases as Aspects – An Approach to Software Composition, AspectProgramming Inc