KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

NESNEYE YÖNELİK SİSTEMLERDE UYUM VE SINIF İÇİ BAĞIMLILIĞIN ÖLÇÜLMESİNDE YENİ BİR YAKLAŞIM

DOKTORA TEZİ

Bilgisayar Yük. Müh. Özcan KURUBAŞ

Anabilim Dalı: Elektronik ve Haberleşme Mühendisliği

Danışman: Doç. Dr. Nevcihan DURU

KOCAELİ, 2011

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

NESNEYE YÖNELİK SİSTEMLERDE UYUM VE SINIF İÇİ BAĞIMLILIĞIN ÖLÇÜLMESİNDE YENİ BİR YAKLAŞIM

DOKTORA TEZİ

Özcan KURUBAŞ

Tezin Enstitüye Verildiği Tarih: 6 HAZİRAN 2011

Tezin Savunulduğu Tarih: 9 EYLÜL 2011

Tez Danışmanı	Üye	Üye
Doç.Dr. Nevcihan DURU	Prof.Dr. Oya KALIPSIZ	Prof. Dr. Kadir ERKAN
(.....)	(.....)	(.....)

Üye	Üye
Prof. Dr. Hasan DİNÇER	Yrd. Doç. Dr. Banu DİRİ
(.....)	(.....)

KOCAELİ, 2011

ÖNSÖZ ve TEŞEKKÜR

Yazılım kalite ölçütleri yazılımın kalitesini belirlemede yardımcı nicel veriler üreten bu nedenle çok sık kullanılan yardımcı ölçütlerdir. Özellikle oluşturulan sistemlerin bakımının kolay olmasını ve yeniden kullanılabilir olmasını sağlama açısından uyum ölçütleri yazılım ölçütleri içerisinde büyük önem taşımaktadırlar. Özellikle artan ihtiyaçlar ve oluşturulacak olan sistemlerin büyüklüğü uyum ölçütlerinin önemini yeniden ortaya koymakta ve aktif olarak üzerinde çalışılan bir konu haline getirmektedir. Günümüze kadar sunulan ölçütler oluşturulan ya da oluşturulacak sistemlerin yazılım anlamında uyumunu belirlemede yetersiz kaldıkları görülmektedir. Bu nedenle özellikle sınıf karakteristiklerini ve sınıf üyeleri arasındaki etkileşimleri temel alan, kapsamlı ve çok yönlü ölçütlere gereksinim duyulmaktadır. Ancak bu şekilde oluşan ihtiyacı karşılayacak ve etkin sonuçlar ile ilgili kişileri yönlendirebilecek uyum ölçütleri geliştirilebilecektir.

Çalışmam süresince gösterdiği emek, destek ve ilgiyle çalışmanın gerçekleşmesi ve ilerlemesini sağlayan danışman hocam Sayın Doç. Dr. Nevcihan DURU'ya,

Çalışmam sırasında fikirleriyle bana yol gösteren ve çalışmanın gerçekleşmesini sağlayan hocam Sayın Prof. Dr. Oya KALIPSIZ'a,

Çalışmam sırasında fikirleriyle bana yol gösteren, destekleyen ve yeni ufuklar açan hocam Sayın Prof. Dr. Kadir ERKAN'a,

Çalışmam sırasında desteğini hiçbir zaman esirgemeyen eşim Evrim Orhan KURUBAŞ'a,

Eğitim ve kariyer hayatım süresince sevgi ve emeği ile her zaman yanımda olan anneme ve babama,

Sonsuz teşekkürlerimi sunuyorum.

Bilgisayar Yük. Müh. Özcan KURUBAŞ

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER

c	: herhangi bir sınıf
m	: herhangi bir metot
a	: herhangi bir öznitelik
A(c)	: c sınıfı öznitelik kümesi
M(c)	: c sınıfı normal metot kümesi
I(c)	: c sınıfı etkileşim sayı kümesi
$A_D(m)$	: m tarafından doğrudan erişilen öznitelik kümesi
$A_I(m)$	: m tarafından dolaylı olarak erişilen öznitelik kümesi
$A_{DW}(m)$	: m tarafından doğrudan yazılan öznitelik kümesi
$A_{DR}(m)$	: m tarafından doğrudan okunan öznitelik kümesi
$A_{IW}(m)$	: m tarafından dolaylı yazılan öznitelik kümesi
$M_D(a)$	: a'ya doğrudan erişen metot kümesi
$M_I(a)$	: a'ya dolaylı erişen metot kümesi
$M_{DR}(a)$	: a'yı doğrudan okuyan metot kümesi
$M_{IR}(a)$	: a'yı dolaylı yazan metot kümesi
$A_{DA}(a)$	: a kullanılarak elde edilen öznitelik kümesi
$I_{MA}(m,a)$	: m'nin a'ya erişim sayısı
$M_{DC}(m)$	: m tarafından doğrudan çağrılan metot kümesi
$M_{IC}(m)$	: m tarafından dolaylı çağrılan metot kümesi
$G_{MA}(c)$	: c sınıfı üyeleri arasındaki etkileşimin grafik gösterimi
$G_{AA}(c)$	: öznitelikler arasındaki etkileşimin grafik gösterimi
V	: G grafiğindeki köşe
E	: G grafiğindeki kenar
I	: G grafiğindeki köşe ve kenar arasındaki etkileşim
$M_{IP}(m,a)$	: m'den a'ya erişmek için çağrılan metot kümesi
IP(m,a)	: M_{IP}'teki her bir yol
$M_{SP}(m,a)$	: m'den a'ya en kısa IP
WM(m)	: metot ağırlığı
WA(a)	: öznitelik ağırlığı
$M_{CC}(m)$	: m'nin genel uyuma katkısı
$C_{CD}(c)$	: c'nin içinde bulunduğu duruma göre en yüksek uyum değeri
BCOM(m)	: temel metot uyumu
BCOM*(m)	: m için ayrık grup etkisi
$D_G(c)$	: c içindeki ayrık grup kümesi
COM(m)	: metot uyumu
ES	: erişim sayısı
ET	: erişim türü
YD	: yazma durumu
Co	: connectivity
Coh	: cohesion

Alt indisler

i,j	: sınıf içindeki metot ya da öznitelik

Kısaltmalar

LCOM	: Uyum Azlığı (Lack Of COhesion)
TCC	: Sıkı Sınıf Uyumu (Tight Class Cohesion)
LCC	: Gevşek Sınıf Uyumu (Loose Class Cohesion)
COC	: Sınıf Uyumu (Class of Cohesion)
PCA	: Temel Bileşen Analizi (Principal Component Analysis)
SRP	: Tek Sorumluluk Prensibi (Single Responsibility Principle)
IEEE	: Elektrik Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers)
ROC	: İlişki Tabanlı Uyum (Related Oriented Cohesion)
SOM	: Kendi Kendini Düzenleyen Harita (Self Organizing Map)
PC	: Temel Bileşen (Principal Component)

NESNEYE YÖNELİK SİSTEMLERDE UYUM VE SINIF İÇİ BAĞIMLILIĞIN ÖLÇÜLMESİNDE YENİ BİR YAKLAŞIM

Özcan KURUBAŞ

Anahtar kelimeler: Yazılım Mühendisliği, Sınıf Uyumu, Ölçüt Geliştirme, Sınıf Üyeleri Arası İlişkiler.

Özet: Nesneye yönelik sistemlerde, sınıf uyumu (cohesion) sınıf üyeleri arasındaki ilişkinin derecesini ifade etmektedir. Sınıf uyumu kötü tasarlanmış sınıfları geliştirmek için kullanılabilmektedir. Geçmiş yıllarda, literatürde, metotların öznitelik kullanım kriterini temel alan çeşitli ölçütler önerilmiştir. Bu ölçütler sınıf uyumunu sınıf üyelerinin arasındaki bağlantı temelinde elde etmişlerdir. Bununla birlikte, sunulan bu ölçütlerin hiçbiri üyeler arasındaki ilişkileri tam olarak tanımlamamışlardır. Özellikle, metot-öznitelik referanslarının ilişki yönü, metot-öznitelik referans sayıları ve öznitelikler arasındaki bağımlılıklar göz ardı edilmiştir. Bu tür problemleri gidermek için bu çalışma kapsamında sınıf üyeleri arasındaki tüm etkileşim desenlerine odaklanan yeni bir metot tabanlı ölçüt önerilmektedir. Önerilen ölçüt, bir matematiksel çerçeve kullanılarak analitik olarak doğrulanmaktadır. Ek olarak, önceden tanımlı sınıflar üzerinde ve bir açık kaynak kodlu uygulama üzerinde önerdiğimiz ölçütün etkinliğini göstermek için durum çalışması yapılmıştır. Daha sonra, elde edilen sonuçların karşılaştırılması ile önerdiğimiz ölçütün mevcut ölçütlerin elde edemediği farklı bir yaklaşıma sahip olduğunu gösterilmiştir.

A NEW APPROACH ABOUT THE MEASUREMENT OF THE CLASS COHESION AND THE CLASS INNER DEPENDENCY FOR THE OBJECT ORIENTED SYSTEMS

Özcan KURUBAŞ

Key Words: Software Engineering, Class Cohesion, Developing Metric, Interactions Between Class Members.

Abstract: In object-oriented systems, cohesion refers to the degree of the relatedness of the class members. Cohesion could be used to improve poorly designed classes. In previous years, several metrics based on instance variable usage criteria for methods have been proposed in the literature. They capture class cohesion in terms of connections among class members. However, almost none of these metrics have completely defined the relationships among the members. Specially, the relation direction of the method-instance variable references, the number of method-instance variable references, and dependencies among instance variables are neglected. In order to eliminate these problems, this study proposes a new method-based metric which focuses on all interaction patterns among the class members. The proposed metric is analytically verified using a mathematical framework. Additionally, a case study on predefined classes is performed to demonstrate the effectiveness of our new metric. Then, by comparing results, we demonstrate that proposed metric captures a different approach which is not captured by the existing cohesion measures.

1. GİRİŞ

Yazılımlar modern günlük yaşantımızın içinde çok önemli bir yere sahiptirler. Birçok durumda, yazılım sisteminde oluşabilecek sorunlar, ekonomik kayıp veya insan hayatı kaybı gibi kötü sonuçlara sebep olabilirler. Bu nedenle, yazılımın kalitesini belirleyebilmek çok önemlidir. Daha kaliteli sistemlerin inşa edilmesi, son yıllarda tüm yazılım mühendisliği çabalarını yönlendiren amaç olarak karşımıza çıkmaktadır. Bir sistem, anahtar kalite belirleyicilerini destekleyebilmeli ve sahip olacağı akıllı matematiksel modeller yardımı ile kaliteyi kontrol altında tutabilmelidir.

Nesneye yönelik diller ve geliştirme ortamları kaliteli sistemlerin yapımı için imkân sağlama açısından oldukça fazla yol kat etmişlerdir. Fakat bu noktada bir problem karşımıza çıkmaktadır. Günümüzde eskiye nazaran istekler ve geliştirilen sistemlerden beklentiler artmıştır. Geliştirilen donanımlar bu beklentileri karşılayacak düzeydedir. Bunun yanında geliştirilen yazılım sistemlerinin de yazılım fonksiyonelliği açısından beklentileri karşılaması zorunluluk haline gelmiştir. Bu durum ise yazılım sistemlerinin daha karmaşık ve kontrolünün daha zor olmasına sebep olmaktadır. Sonuç olarak; birçok nesne ve nesneler arası birçok ilişki içeren nesneye yönelik olarak geliştirilen sistemlerde, karmaşıklık hızlı bir şekilde artış göstermektedir. Ortaya çıkan bu karmaşıklığı daha rahat yönetebilmek için ise bir çözüm gereklidir. Çözüm, nesneye yönelik dil ve geliştirme ortamlarına eklenebilecek ve kullanıcılara yol gösterebilecek olan ölçütlerin oluşturulmasıdır. Böylece bu ölçütler sistemlerin uygun şekilde tasarımında ve geliştirilmesinde yazılımcılara yardımcı olacaktır. Bu tür ölçütler, yazılım geliştirme sürecinde ne durumda olduğumuz hakkında bizi uyaracak bilgiler üretebilecek, nesneler arasındaki ilişkilerin durumu ve gidişatı hakkında önemli bilgiler sağlayabilecektir.

Yazılım ölçütleri bu önemli görevleri nedeni ile yazılım mühendisliğinin birçok disiplini için ihtiyaç haline gelmiştir [1]. Yazılım kalitesi alanında, ölçütler cohesion (uyum), coupling (bağımlılık) ve complexity (karmaşıklık) gibi çeşitli yazılım

özelliklerine değer biçmek için kullanılmaktadırlar. Yazılım ölçütleri, yazılım geliştirme sürecini tahmin etmede ve üretilen yazılımın kalitesini değerlendirmede nicel bir anlam sunmaktadır. Kalite ölçütleri potansiyel faydalarını yazılımların tasarlanmasında, yazılım süreci içerisinde yol göstericilik yaparak, yazılımları daha anlaşılır ve bakımlarının daha kolay olmasını sağlayarak göstermektedir. Yapılan çalışmalar, yazılım ölçütleri ile temel yazılım kalite özellikleri olan hataya meyilli olma ve bakım kolaylığı çabası arasında ilişki olduğunu açık olarak ortaya koymaktadır [2,3].

Tezin kapsamını oluşturan uyum, bir bileşen içindeki üyelerin birbirleri ile ilişki derecesine işaret etmektedir. Yüksek uyum, bileşenler açısından arzu edilen bir özelliktir. Yüksek uyuma sahip bileşenlerin daha fazla bakım kolaylığı ve daha fazla tekrar kullanıma meyilli olduğu geniş bir kabul görmektedir [4,5]. Allen ve Khoshgoftaar uyumu modüller arası bağımlılık derecesi olarak tanımlamışlardır [6]. Fenton'a göre, bir modülün uyumu, modüle ait bağımsız bileşenlerin aynı görevi yerine getirmeye gereksinim duymalarının büyüklüğüdür [7]. Yourdon ve Constantine uyumu, geleneksel uygulamalar için fonksiyonel ilişkilerinin büyüklüğü olarak tanımlamışlar ve uyum için sıralı ölçekte bir sınıflandırma yapmışlardır [8]. Bu sıralı ölçek şu şekilde oluşturulmuştur:

- Tesadüfî uyum (Coincidental cohesion): Bir modülün parçaları rastgele ve keyfi bir şekilde bir araya getirilmiştir, parçaların arasında önemli bir ilişki bulunmamaktadır. Örneğin, matematik fonksiyonlarını bir araya getiren modül gibi. Bu durum en kötü durumdur.

- Mantıksal uyum (Logical cohesion): Bir modülün parçaları mantıksal olarak benzer kategoride işler yaptıklarından dolayı bir araya getirilmişlerdir. Örneğin, tüm I/O rutinlerinin bir araya getirilmesi gibi.

- Geçici uyum (Temporal cohesion): Bir modülün parçaları işlenecekleri zaman, bir araya gelen parçalardan oluşmaktadır. Parçalar programın belirli bir zaman diliminde işletilen parçalarından oluşmaktadır. Örneğin, hata alındıktan sonra

çağrılan bir fonksiyonun, dosyaları kapaması, kütüğe yazması ve kullanıcıları uyarması bu türden uyumu göstermektedir.

- Prosedürel uyum (Procedural cohesion): Bir modülün parçaları belirli bir işletim sırasını takip eden parçalardan oluşmaktadır. Örneğin, dosya haklarını kontrol eden ve dosyayı açan bir fonksiyon, bu türden bir uyumu göstermektedir.

- İletişimsel uyum (Communicational cohesion): Bir modülün parçaları aynı veri üzerinde işlem yapmaktadır. Örneğin, aynı kayıt bilgisi üzerinde işlem yapan bir modül gibi.

- Sırasal uyum (Sequential cohesion): Bir modülün parçaları bir birleştirme hattındaki gibi, birinin çıkışı diğerinin girişi olacak şekilde bir arada bulunmaktadır. Örneğin, dosyadan veri okuyan ve bu veriyi işleyen bir fonksiyon bu türden uyuma örnektir.

- Fonksiyonel uyum (Functional cohesion): Bir modülün tüm parçaları iyi tanımlı tek bir görevi yerine getirmek için bir arada bulunmaktadır. Örneğin, bir açının sinüs değerinin hesap edilmesi gibi.

Biemann ve Ott uyumu, veri dilimleri şeklinde tanımlamışlar ve fonksiyonel uyumu tanımlayan ilk kişiler olmuşlardır [9]. Veri dilimi, bir program değişkenini etkileyen program kodu parçasını içermektedir. Uyum hesabı için değişkenleri etkileyen bu kod parçaları tespit edilmektedir, sonrasında ise bu kod parçalarından, tüm değişkenleri etkileyen kod satırları yapıştırıcı olarak kabul edilerek oransal bir birliktelik değeri elde edilmektedir.

Nesne tabanlı değerler dizisi içinde, sınıf uyumu sınıf üyeleri arasındaki ilişkililiğin ölçüsü olarak düşünülmektedir [10]. İlişkililiğin sınıf açısından anlamı, sınıf tarafından sunulan metotların benzerliğidir. Eğer bir sınıfın metotları benzer fonksiyonellik sunuyorsa sınıf uyumludur, benzer olmayan fonksiyonellik sunuyorsa sınıf az uyumludur ya da hiç uyumlu değildir. Bir sınıf, metotları yüksek derecede karşılıklı olarak ilişkili ise uyumludur. Rumbaugh, yüksek fonksiyonel uyumu, bir

bileşen elemanlarının tümünün iyi tanımlanmış ve sınırları belirlenmiş davranışı sağlamak için bir arada çalışmasının varlığı olarak tanımlamıştır. Bir sınıf, birbirleri ile ilişkisiz üyeler kümesi olmamalıdır, sınıfın üyeleri ilgili nesnelerin bazı davranışlarını sağlamak için birlikte çalışmalıdırlar [11]. Grady Booch ise düşük uyumu olan bir sınıfı uygunsuz ya da rastgele bir soyutlamaya sahip bir sınıf olarak tanımlamaktadır [12].

Literatürde, sınıf uyum değerini ölçmek için çeşitli ölçütler önerilmiştir. Bugüne kadar sunulan ve kategorize edilen başlıca uyum ölçütleri [5]'de yer almaktadır. Ölçütler, metotlar tarafından kullanılan örnek değişken sayılarını ya da örnek değişkenleri paylaşan metot sayılarını hesaplamışlardır. Bu ölçütler, bir sınıf içindeki üyeler arasındaki bağlantılar anlamında sınıf uyumunu elde etmişlerdir. Bunların çoğu bugüne kadar denenmiş ve geniş bir şekilde literatürde tartışılmıştır [2, 13].

Birçok araştırmacı sınıf uyumunu, önerdikleri kendi ölçütleri ile tanımlamışlardır. Önerilen ölçütlerin birçoğu, Chidamber ve Kemerer tarafından tanımlanan LCOM (Lack Of COhesion) ölçütünden esinlenmiştir [14]. Literatürde görüleceği gibi birçok araştırmacı, LCOM ölçütünü tekrar tanımlamıştır. Bir sınıf, örnek değişkenlerinden çoğu metotları tarafından kullanılıyorsa [5, 15] ya da çok sayıda metot çifti örnek değişkenleri paylaşıyorsa, çok uyumludur [10, 13, 14, 16, 17].

Chidamber ve Kemerer, sınıf uyumuna değer biçmek için LCOM (LCOM1 ve LCOM2) ölçütünü önermişlerdir. LCOM1, ortak öznitelik kullanmayan metot çiftlilerinin sayısıdır. LCOM1'in tekrar tanımlanması ile oluşturulan LCOM2, ortak öznitelik kullanmayan metot çiftlileri sayısından en azından bir ortak öznitelik paylaşan metot çiftlilerinin sayısının çıkarılması ile hesaplanmaktadır ve literatürde oldukça geniş olarak tartışılmıştır [5, 15, 17]. LCOM2, eğer değeri negatif ise 0'a eşit olduğu kabul edilmektedir.

Li ve Henry, LCOM'u tekrar tanımlamışlardır [3]. LCOM3 olarak tanımladıkları ölçüt, metotların ayrık kümelerinin sayısıdır. Her bir küme en azından bir öznitelik paylaşan metotları kapsamaktadır.

Hitz ve Montazeri LCOM3'ü tekrar tanımlamışlardır [17]. Bu ölçüt, grafik teorisi tabanlıdır ve bir grafikteki bağlı bileşenlerin sayısı olarak tanımlanmıştır. Eğer metotlar en azından bir öznitelik paylaşıyorlarsa bağlıdırlar. Grafikteki köşeler, sınıfın metotlarını temsil etmektedir. Eğer ilişkin metotlar aynı nesne değişkenine erişiyor ise, iki köşe arasında bir kenar yer almaktadır.

LCOM3'ü takiben, Hitz ve Montazeri LCOM4'ü tanımlamışlardır [17]. LCOM3'ten farklı olarak öznitelik paylaşmanın yanında metot işletimleri de metotları bağlı hale getirmektedir. Bundan dolayı grafik üzerinde köşeleri ifade eden metotlar arasında metot işletimleri dolayısıyla da kenar çizilmektedir.

Hitz ve Montazeri, LCOM4 yanında Co (Connectivity) olarak adlandırılan ölçütü tanımlamışlardır. Co, LCOM4 uyum değeri bir olduğu zaman işletilmektedir. Bir başka deyişle, sınıf üyeleri bir şekilde ilişkili iseler uygulanabilmektedir. Co, grafik üzerinde yer alan kenarları ve köşeleri işleyerek uyum değerini 0 ile 1 arasına eşleştiren bir ölçüttür.

Sonrasında Hendersen-Sellers LCOM5'i önermişlerdir [15]. LCOM5, başvurulan öznitelik sayısı tabanlıdır. Bir sınıf, kendi nesne değişkenlerinin büyük bir kısmına kendi metodu tarafından başvuruda bulunuyorsa, daha uyumludur.

Briand LCOM5 için tekrar bir tanımlama sunmuştur [5]. Yeni tanımlama literatüre Coh olarak geçmiştir. LCOM5'ten farklı olarak bazı özniteliklerin metotlar tarafından erişilmemesi durumunu göz önüne almaktadır.

Bieman ve Kang, TCC (Tight Class Cohesion) ve LCC (Loose Class Cohesion) ölçütlerini uyum ölçütleri olarak sunmuşlardır [4]. Onlar da ortak değişken kullanan metot çiftlerini kullanmışlardır. Bir nesne değişkeni, bir metot tarafından doğrudan ya da dolaylı olarak kullanılabilir. Eğer bir nesne değişkeni, Mi (herhangi bir i. metot) metodunun gövdesinde bulunuyor ise, bu nesne değişkeni Mi metodu tarafından doğrudan kullanılmaktadır. Eğer bir nesne değişkeni, Mj tarafından doğrudan kullanılıyor ise ve Mj'de Mi tarafından doğrudan ya da dolaylı olarak işletiliyorsa, bu nesne değişkeni Mi metodu tarafından dolaylı olarak

kullanılmaktadır. Eğer iki metot, doğrudan ya da dolaylı olarak ortak bir değişken kullanıyor iseler, bu iki metot doğrudan ilişkilidir. TCC, doğrudan ilişkili metot çiftlerinin göreli oranı olarak tanımlanmaktadır. LCC, doğrudan ya da dolaylı olarak ilişkili metot çiftlerinin göreli oranı olarak tanımlanmaktadır.

Bunların dışında, sınıf içindeki diğer öznitelikler üzerinden hesaplanan bağımlı özniteliklerin etkisini katarak mevcut ölçütleri geliştirmeyi hedefleyen çalışmalar [18, 19] içinde yer almaktadır. Bağımlı öznitelik değeri diğer öznitelikler tarafından belirlenebilen öznitelikleri ifade etmektedir. Bu tür çalışmalar ölçüt önermekten ziyade teorik olarak farklı bir bakış açısını yansıtmaktadırlar.

Sınıf içindeki uyumlu kısımların boyutları üzerinde yapılan çalışma [20] içinde yer almaktadır.

Bağımlılık analizine dayanan uyum belirleme çalışması örneği [21] içinde yer almaktadır. Bu tür çalışmalarda çeşitli kurallar kapsamında üyeler arasındaki bağımlılığın tanımlanması ve buna göre sınıf uyumunun elde edilmesini hedeflemektedir.

Kavramsal uyum ölçütlerine dair örnek çalışmalar [22, 23] içinde yer almaktadır. Kavramsal uyum, sınıf üyelerinin kavramsal olarak incelenerek benzer anlamlar taşıması durumuna bir değerleme yapılması ile elde edilmektedir.

İstemci tabanlı uyum ölçütleri, sınıfa yapılan dış çağrımları temel alarak uyumu elde etmeyi hedefleyen ölçütlerdir. Buna göre sınıfa yapılan çağrımlar dinamik olarak çalışma zamanında değerlendirilmektedir. Bu tür çalışmaya örnek [24] içinde yer almaktadır.

Metotlar arasındaki uyumu işleyen çalışma [25] içinde yer almaktadır. Bu ölçüt parametre görülme matrisi kullanmaktadır, buna göre her bir metot için bir satır ve her bir veri tipi için bir kolon açılır. Metot ilgili tipte parametre kullandığında bu hücreler için 1 değeri konur diğer değerler 0 olarak korunur. İşlemler bu matris için yapılır.

Bağımlılık matris tabanlı uyum çalışması [26] içinde yer almaktadır. Bağımlılık matrisi bir sınıf içindeki metotlar ve öznitelikler arasındaki bağımlılığın ilişki derecesini gösteren bir matristir.

Her bir metot çifti arasındaki kesişimin çözünürlüğü hususunda yapılan uyum ölçütü çalışması [27] içinde yer almaktadır.

Özellik kullanım ve metot işletim kıstaslarını inceleyerek TCC ve LCC ölçütlerini geliştiren bir çalışma [28] içinde yer almaktadır. Yapılan bu çalışma içerisinde de özel metotlara nasıl davranılacağı, öznitelikler arasındaki ilişkiler ve metot-öznitelik etkileşim sayıları göz önüne alınmamıştır. Metot ve öznitelik arasındaki ilişki bir defa ortaya çıktığında ilişki yoğunluğundaki artış ya da azalışa ölçüt tepki verememektedir.

Yüksek seviyeli tasarım tabanlı uyum ölçütlerine örnek çalışma [29] içinde bulunabilir. Bu tür ölçütler yapılan tasarımın başında uyumu belirlemeye çalışmaktadır. Bu şekilde kodlama uygun şekilde yapılabilmektedir. Fakat bu tür çalışmalar gerçekçi olmayan varsayımlara dayanmakta ve yeteri kadar iyi bir şekilde modeli üretilememektedir.

Mevcut ölçütler üzerinde yapılan matematiksel geçerleme çalışması [30] ve bağlantılar açısından mevcut ölçütlerin değerlendirme çalışması [31], yukarıda bahsi geçen kabul görmüş bazı ölçütleri farklı açılardan değerlendirmektedir. Fakat bu çalışmalardan önce yapılmış olan [32, 33] içinde yer alan çalışmalar mevcut ölçütlerin bağlantılar açısından ve matematiksel açıdan eksik olduklarını açık olarak ortaya koymaktadır.

Ayrıca son zamanlarda bu alanda birçok çalışma yapılmıştır, yapılan bu çalışmalar [34, 35, 36] içinde yer almaktadır.

Sınıf uyum ölçütleri yazılım sürecinin birçok aşamasında fayda sağlamaktadırlar. Bu aşamalara örnek vermek gerekirse; tasarım kalitesine değer biçmede [37, 38],

yazılım kalite ve hata eğilimi tahmininde [39, 40], yazılımın modülerleştirilmesinde [41, 42], tekrar kullanılabilecek bileşenlerin belirlenmesinde [43, 44] kullanılabilmektedirler.

Bugüne kadar uyum ölçütü geliştirme hususunda birçok araştırma yapıldığı halde hala genel olarak kabul edilmiş tanımlar ve ölçütler bulunmamaktadır. Fenton ve Pfleeger [7] içindeki çalışmasında bunu göstermişlerdir. Bu çalışmadan sonra yapılan çalışmalar da ise özellikle üzerinde fazlaca tartışılan mevcut ölçütler belirlenmiş çalışmalar bu ölçütler üzerinden sorgulanır hale gelmiştir [5, 38, 45].

1.1. Araştırma Problemi

Uyum, bir yazılımın kalitesini değerlendirmede önemli bir özellik olarak kabul edilmektedir. Uyum nesneye yönelik bir sistemin kalitesini, bakılabilirliğini ve yeniden kullanabilirliğini büyük ölçüde etkilemektedir. Fakat mevcut araştırmalar ve önerilen sınıf uyumu ölçütleri sınıf uyumunu etkin ve verimli şekilde belirlemek için yeterli değildirler. Uyum ölçütleri, sınıf karakteristiklerini iyi bir şekilde yakalamakta ve bu karakteristikleri uyum hesaplama sürecine katma hususunda yetersiz kalmaktadırlar. Özellikle, sınıf üyeleri arasındaki ilişkilerin tam olarak yakalanması, ilişki tiplerinin hesaplama sürecine katılması ve ayrık grupların uyuma etkisinin yansıtılması hususunda yeterli etkinliğe sahip değildirler. Bundan dolayı, mevcut ölçütlerin sınıf uyumunu belirleme açısından takip ettikleri yollar ve elde ettikleri sonuçlar memnun edici düzeyde değildir.

1.2. Tez Yaklaşımı

Kaliteli sistemlerin inşa edilmesi, son yıllarda tüm yazılım mühendisliği çabalarını yönlendiren amaç olarak karşımıza çıkmaktadır. Çünkü yazılım sistemlerinde meydana gelebilecek hatalar çok büyük maliyetlere sebep olabilmektedir. Günümüzde kullanılan nesneye yönelik geliştirme ortamları kaliteli sistemler sağlamak açısından oldukça fazla yol kat etmişlerdir. Fakat günümüzde yazılımlardan beklentiler arttığı için bu gelişme yeterli olmamaktadır. Bu açıdan bakıldığında kendimize sorduğumuz soru şudur:

“Beklentilerin arttığı bu ortamda sınıf uyumunu belirleyebilmek için mevcut olan ölçütlerden farklı olarak sınıf karakteristiklerini daha etkin bir şekilde yakalayabilen, bunları uyum sürecine uygun bir şekilde katabilen ve daha verimli ölçümler yapabilecek bir ölçüt geliştirebilir miyiz ?”.

Bu noktadan hareketle sahip olduğumuz motivasyon, çeşitli sınıf karakteristiklerini dikkate alan, günümüz gelişen ihtiyaçlarını karşılayacak, mevcut ölçütlerin eksikliklerini giderecek, daha etkin ve verimli bir ölçütün geliştirilmesini sağlamaktır.

1.3. Tezin Katkısı

Yukarıda bahsi geçen bazı ölçütler, uyumu farklı bir değer beklendiği halde minimum olarak hesaplamakta, bazıları ise üyeleri bir şekilde ilişkili sınıfa daima maksimum uyum değerini atamaktadırlar. Hâlbuki sınıflar farklı ilişki desenlerine sahip olabilmektedirler. Ayrıca, sınıf içinde, farklı fonksiyonellik sunan gruplar ortaya çıkabilmektedir. Uyum sınıfın tek bir fonksiyonellik sunmasına dayanmaktadır. Fakat mevcut ölçütler, sınıf farklı fonksiyonellik sunan gruplar içerse de sınıf uyumunu yüksek olarak ölçmektedirler. Hâlbuki bu istenmeyen bir durumdur. Bunların yanında dikkat çeken bir diğer husus ise metotların öznitelikleri kaç kez kullandığını ifade eden etkileşim sayılarının uyum sürecine katılmamasıdır.

Bir sınıf, üyeleri arasındaki etkileşimler sonucu bir fonksiyonellik sunmaktadır. Sınıfın fonksiyonelliği temel olarak metotlar üzerinden sunulmaktadır. Üyeler arasındaki etkileşimlerin yoğunluğu sınıfın uyumlu, tek bir amaca yönelik fonksiyonellik sunduğunu göstermesi açısından önemlidir. Bunun yanında, sınıf uyumu kendini dışarıya metotlar üzerinden göstermektedir. Öznitelikleri bir amaç etrafında bir araya getiren ve onları kullanan metotlardır. Sınıf uyumu açısından metotlar bu işlevleri ile önemli bileşenlerdir. Metotlar ise uyumluluğu öznitelikleri kullanmaları ile göstermektedirler. Bundan dolayı, metot ve öznitelik, sınıf ve sınıf uyumu için iki temel bileşen olarak karşımıza çıkmaktadır. Sınıf uyumu ise temel bileşenler arasındaki etkileşimler sonucu oluşmaktadır. Etkileşimler, temel etkileşim

olan metot-öznitelik etkileşimleri, metot-metot etkileşimleri ve öznitelik-öznitelik etkileşimleridir. Metot-öznitelik etkileşimleri, erişim, etkileşim sayıları, etkileşim tipleri ve etkileşim deseninden meydana gelmektedir ve temel uyumu oluşturmaktadır. Bir metot, metot-metot etkileşimleri ile doğrudan erişemediği özniteliklere bu şekilde erişme fırsatı bulabilmektedir. Öznitelik-öznitelik etkileşimleri ile bağımlı öznitelik etkisi sürece katılabilmektedir.

Bazı metotlar sınıf uyumuna istenildiği şekilde katkıda bulunamazlar. Bu özel metotlar bazı özel işleri yerine getirmek için kullanılırlar ve sadece bazı özniteliklere erişirler. Yapıcı, yıkıcı, erişim ve delegasyon metotları bu tür metotlardır. Eğer bu tür metotlar üzerinde işlem yapılmaz ya da uyum elde etme süreci dışında bırakılmaz iseler uyumu kötü yönde etkileyebilmektedirler. Literatürde, bu tür metotlara çeşitli şekillerde davranılmıştır. Örneğin, bazı araştırmacılar özel metotları uyum sürecine katmış [46], bazıları sadece erişim metotlarını ve yapıcıları dikkate almış [5] ve bazıları ise yapıcı ve yıkıcı metotları uyum sürecine katmıştır [4].

Bu çalışma kapsamında, yapıcı ve yıkıcı metotlar uyum sürecine katılmamıştır. Bu şekilde uyumun yapay bir şekilde artması veya azalmasının önüne geçilmiştir. Erişim ve delegasyon metotları üzerinde ise bazı düzenleme işlemleri yapılmıştır. Buna göre, özel metotların eriştikleri öznitelikler, özel metotları doğrudan çağıran metotlara geçirilmektedir. Bu noktadan sonra çalışma boyunca uyum sürecine katılan metotlar, özel metot olmayan ve üzerinde düzenleme yapılmış metotlar olarak kabul edilmektedir.

Literatürde yer alan ölçütler metotların doğrudan ya da dolaylı olarak özniteliklere erişmesi üzerine kuruludurlar. Fakat üyeler arasındaki etkileşimin sadece metotların özniteliklere bir şekilde erişip erişmediği şeklinde yorumlanması uyumun sonucunu kötü yönde etkileyebilmektedir. Bu etki uyumun uygun şekilde elde edilememesine sebep olmaktadır. Bundan dolayı, metotların özniteliklerle olan etkileşim sayılarının ve etkileşim tiplerinin (okuma ve yazma) uyum elde etme sürecine katılması gerekli bir durum olarak ortaya çıkmaktadır. Gerçekte uyum ölçütü, sınıf uyumunu belirlerken erişim durumu, etkileşim sayısı, etkileşim tipi ve etkileşim desenini dikkate almalıdır. Sınıf uyumu bu bilgilerin değişimi ile hesap edilmelidir. Böylece

sadece erişimin olup olmadığı durumundaki uyumdan daha net uyum değerlerinin elde edilmesi sağlanabilmektedir. Etkileşim sayısının artması tek bir erişim sayısı üzerinden yapılan uyum hesabının daha ayrıntılı hale gelmesini sağlamaktadır. Ayrıca bir özniteliği değiştiren (yazan) bir metot, o özniteliği kullanan (okuyan) metotları etkileyeceğinden dolayı, metotlar arasında dolaylı olarak bir etkileşim oluşmaktadır. Bu etkileşim ise metot uyumunu arttırıcı bir etki olarak karşımıza çıkmaktadır. Bunun yanında, ardışık iki uyum değeri arasındaki geçiş değerleri de doldurulmakta ve daha ayırt edici uyum değerleri elde edilebilmektedir. Sadece erişim temelinde hesaplanan sınıf uyum değerinde, metot uyumları bu genel uyum değerinden erişim sayıları eşit olduğu sürece aynı değeri almaktadırlar. Etkileşim sayılarının ve erişim tiplerinin sürece katılması ile metotlar aynı erişime sahip olsalar bile etkileşim sayıları ve etkileşim tipleri farklı olduğu sürece farklı uyumlara sahip olacaktır. Uyumu etkileyen faktörlerin artması metotların uyumları arasındaki benzerliği azaltmakta ve metotların uyum değerlerinin daha ayırt edici hale gelmesini sağlamaktadır. Böylece metotların sınıf uyumuna etkileri değişmekte ve sınıf uyumunu bozan metotlar daha kolay yakalanabilmektedir.

Ayrıca sınıf içerisinde bazen bir özniteliğin değeri doğrudan ya da dolaylı olarak diğer öznitelikler tarafından belirlenebilmektedir [18]. Bu tür özniteliklere önerilen yaklaşım kapsamında bağımlı öznitelikler adı verilmektedir. Böyle bir durum oluştuğunda öznitelik, diğer özniteliklere akış ya da kontrol bağımlı hale gelmektedir. Bağımlı özniteliklerin sınıf uyumu açısından etkisi, onları kullanan metotların uyumu üzerinde olan etkileri ile ortaya çıkmaktadır. Kendine yoğun şekilde bağımlı olunan öznitelikleri kullanan metotlar diğerlerine göre daha fazla uyumlu hale gelmektedir. Çünkü bağımlı olunan öznitelik, bir anlamda kendi içinde diğer öznitelikleri de barındırmaktadır.

Bu çalışmada, erişim durumu, etkileşim sayısı, etkileşim tipi ve etkileşim deseni önerilen yaklaşım ile ölçüm sürecine katılmaktadır. Önerilen yaklaşım daha önce tarafımızdan bir sempozyumda ortaya konmuş olan COC (Class of Cohesion) [32] ölçütünü temel almaktadır. Bu çalışma tez kapsamında ilk zamanlarda geliştirilmiş olan ölçüte dair çalışmayı içermektedir. COC ölçütü kendi içinde erişim ve etkileşim deseni değişkenlerine sahiptir, fakat etkileşim sayıları ve etkileşim tiplerini dikkate

almamaktadır. Önerilen ölçüt bu eksikliği gidermekte ve daha hassas uyum değerleri elde edilmesinde yardımcı olmaktadır.

Birçok mevcut ölçütte olduğu gibi önerdiğimiz ölçüt, kaynak kodun yapısal görünümünü temel almaktadır. Önerilen ölçüt sınıfa yapısal açıdan bakarak sınıf içindeki üyelerin birbirlerine ne derecede ait olduklarını yakalamaya çalışmaktadır. Bu ilişkileri yakalarken yukarıda bahsedilen nesneye yönelik değerler dizisi içinde yer alan sınıf karakteristiklerini dikkate almaktadır ve özellikle mevcut ölçütlerin eksik olan yanlarına daha fazla odaklanmaktadır.

Önerilen ölçüt, içinde barındırdığı bu özellikler ile mevcut ölçütlerden farklı olarak tanımlanmıştır. İçerisinde diğer ölçütler bulunmayan farklı yaklaşımlar bulunmaktadır. Mevcut ölçütlerden farklı olan bu yaklaşımı gösterebilmek için durum çalışması yapılmıştır. Elde edilen sonuçların değerlendirilmesi sonucu, sunulan ölçütün daha fazla çeşitlilikte uyum değerini yakalayabildiğini göstermiştir. Sunulan ölçütün, sınıf üyeleri arasındaki etkileşimin değişikliğine daha fazla duyarlı olması elde edilen bir diğer sonuçtur. Buna göre daha hassas ölçümler elde edilebilmektedir. Mevcut ölçütler belirli durumlarda mantıklı sonuçlar üretebilirken, önerilen ölçüt sınıf içerisindeki yakalayabildiği tüm etkileşimleri işlediği için tüm durumlarda mantıklı sonuçlar üretebilmektedir. Yapılan ikinci durum çalışması ise ölçüm sonuçlarının istatistiksel açıdan incelenmesi içermektedir. Mevcut ölçütlerin sonuçları ile sunulan ölçüt ile elde edilen sonuçlar analize tabi tutulduklarında ölçütün kendine has bir veri seti olduğunu ve diğer ölçütlere göre farklı bir yaklaşımı ortaya koyduğu görülmüştür.

1.4. Tezin Organizasyonu

Bu tez 7 bölüm halinde yazılmıştır. Bölüm açıklamaları aşağıda yer almaktadır.

Birinci bölümde, literatür incelemesi yapılmış, araştırma problemi tanımlanmış, tez yaklaşımı aktarılmış ve çözüm aşamaları ana hatları ile verilmiştir. Bu bölümde tezin genel bir tanımı yapılmıştır.

İkinci bölümde, nesneye yönelik sistemlere ait genel tanımlar ve sınıf karakteristiklerine ait temel özellikler anlatılmaktadır. Bu bölümde bahsi geçen sınıf karakteristikleri, önerilen ölçütün temelini oluşturmaktadır.

Üçüncü bölümde, mevcut ölçütler detaylı olarak aktarılmaktadır. Bu bölümde yer alan ölçütler literatürde fazlaca değerlendirilmiş ve kabul görmüş ölçütlerdir. Önerilen ölçüt bu ölçütlerin ışığında geliştirilmiştir.

Dördüncü bölümde önerilen ölçütün matematiksel modeli ayrıntıları ile birlikte adım adım anlatılmaktadır. Matematiksel modelin her bir parçası nedenleri ile birlikte bu bölümde aktarılmaktadır. Modelin oluşturulmasından sonra çalışma teorik olarak iki çatı yardımı ile değerlendirilmektedir.

Beşinci bölümde deneysel çalışmalar verilmektedir. Deneysel çalışma kapsamında öncelikle tek bir sınıfın önerilen ölçüt ile değerlendirilmesi yapılmaktadır, sonrasında hazırlanmış olan 18 sınıf üzerinde mevcut ölçütler ile önerilen ölçüt için değerlendirilme yapılmaktadır ve son olarak açık kaynak kodlu bir yazılım sistemi üzerinde mevcut ölçütler ile önerilen ölçüt için değerlendirilme yapılmaktadır.

Altıncı bölümde, açık kaynak kodlu yazılım sistemi üzerinde mevcut ölçütler ile önerilen ölçüt için yapılan değerlendirme sonuçları verilmektedir. Bu sonuçlar PCA (Principal Component Analysis) analizi ile değerlendirilmektedir. Elde edilen analiz sonuçları da değerlendirmeler ile birlikte bu bölümde verilmektedir.

Tezin son bölümünde ise, genel olarak elde edilen sonuçlar ve gelecek çalışmaları için yararlı olabileceği düşünülen bazı saptamalara yer verilmektedir.

2. GENEL TANIMLAR

2.1. Giriş

Nesneye yönelik sistemlerdeki uyum hakkında konuşabilmek için, bu tür sistemlerin temel bileşenlerini ve aralarındaki ilişkileri iyi seviyede belirlemek gerekmektedir. Nesne yönelimi kavramı veri yapısı ve davranışların kapsüle edildiği bir küme ayrık objeden oluşan yazılım olarak tanımlanır [47]. Nesneye yönelik diller ve sistemler için geniş olarak kullanılan tanımlar [48] içinde yer almaktadır, tez kapsamında da bu tanımlardan yararlanılacaktır. Bir nesneye yönelik programlama dili nesne-sınıf, kalıtım ve çokbiçimlilik kavramını desteklemelidir. Nesneye yönelik sistemler için geliştirilecek olan ölçütlerde bu temel kavramları kapsayacak şekilde geliştirilmelidir. Aksi halde oluşturulacak olan ölçütler bir şekilde sorgulanabilir halde olacaklardır.

Sınıf karakteristikleri yanında, uyum bir kalite ölçütü olduğu için bu alanında iyi anlaşılması gerekmektedir. Çünkü uyum gibi ölçütlerin ortaya çıkış sebebi yazılımın konu olduğu gereksinimlerin kaliteli bir şekilde karşılanmasıdır. Kalitenin bir şekilde sağlanabilmesi için oluşturulan ölçütlerin, kaliteli olma kararının verilebilmesi amacıyla hassas verileri sağlaması gerekmektedir. Değişen durumlar karşısında farklılık oluşturamayan, olası tüm olasılıklara cevap veremeyen ölçütlerin kalitenin sağlanıp sağlanamadığı hakkında yönlendirici cevaplar verebilmesi mümkün değildir. Unutulmaması gereken konu, ölçülemeyen şeylerin kontrol edilemeyeceğidir.

2.2. Nesne-Sınıf

Bir nesne, nesnenin içsel durumunu ifade eden bir küme öznitelikten ve nesnenin dışsal davranışını gösteren bir küme metottan oluşmaktadır. Farklı bir şekilde

anlatmak gerekirse, bir nesne durum ve davranışın tek bir yapı içinde sarmalanmış halidir [11, 49].

Bir nesne sınıfı, bir küme benzer nesnenin durum ve davranışlarını belirleyen bir tür şablon olarak görülebilir. Bir nesne, nesne sınıfının çalışma zamanında oluşturulmuş bir örneğini ifade etmektedir. Nesne sınıfları, nesneye yönelik tasarımın ve geliştirmenin en temel yapıtaşlarıdır. Nesne sınıfı, soyut veri tipi yaklaşımını ifade eden sarmalama (encapsulation) ve sınıf yapısının görünür ve saklı kısımlarını ifade eden bilgi saklama prensibi (information hiding) üzerine kurulmuştur. Grady Booch'a göre sarmalama, yapı ve davranışı oluşturan soyutlama elemanlarının ayrıştırılması işlemidir [12]. Buna göre sarmalama, arayüz ve uygulama kısmının ayrılmasına yardımcı olmaktadır. Görünür kısım, sınıfın tanımı ya da arayüzü olarak isimlendirilir ve bir küme metot tanımından oluşmaktadır. Gizli kalan kısım ise sınıfın uygulaması olarak isimlendirilir ve metot uygulamaları ile öznitelik tanımlarından oluşmaktadır.

2.3. Kalıtım

Bir sınıf diğer sınıflar ile kalıtım ilişkisi (inheritance) ile ilişkide olabilir, bu durumda bir sınıf diğer sınıftan öznitelikleri ve metotları kalıtım yolu ile devralabilmektedir [49]. Bu durumda kalıtılan sınıf süper sınıf ya da ata sınıf olarak isimlendirilmektedir. Kalıtılmış sınıf ise alt sınıf ya da türeyen sınıf olarak isimlendirilmektedir. Kalıtım yolu ile nesne hiyerarşileri oluşturulabilmektedir. Mevcut ölçütler içerisinde kalıtım ayrı bir kavram olarak ele alınmıştır. Çünkü bir sınıf kalıtım ile temel sınıftan metot ve öznitelik alabilmektedir. Fakat yapılan çalışmalar kalıtımın uyum ölçüm sürecine katılmamasının problem oluşturmadığını göstermektedir [34]. Önerdiğimiz ölçüt için işleme katılacak metotların hangi sınıftan geldiği önem arz etmemektedir. Uygulanan süreç sınıf içerisinde bulunan üyelere göre düzenlenmektedir.

2.4. Çokbiçimlilik

Üçüncü özellik ise çokbiçimlilik (polymorphism)'tir. Çokbiçimliliğin olabilmesi için kalıtım zorunlu kuraldır. Kalıtım yanında soyut sınıf ve soyut metot kavramı ortaya çıkmaktadır. Sınıf hiyerarşisi kurulduğunda temel sınıfın bazı durumlarda mantıklı davranışları olmayabilir. Örneğin şekil hiyerarşisinde kare ya da üçgen özellikleri ile bir anlam ifade ederken, temel tip olan şekil bir anlam ifade etmemektedir. Ayrıca bir alan hesabı yapılacaksa şekil temel tipi için bu hesap bir anlam ifade etmemektedir. Bu durumda sınıf ya da alan hesaplama metodu sanal olarak işaretlenirse hem kalıtım hem de çokbiçimlilik sağlanabilmektedir. Çokbiçimliliğin bir tanımı ise aynı isimdeki metodun farklı nesneler üzerinden çağrılabilmesidir.

2.5. Diğer Sınıf Karakteristikleri

Bu temel özelliklerin yanında metot aşırı yükleme, metot ezme ve mesaj aktarımı farklı sınıf karakteristikleri olarak karşımıza çıkmaktadır.

Metot aşırı yükleme, aynı isme sahip fakat farklı sayı ya da farklı tiplerde metot tanımlamaya izin vermektedir [50]. Bu şekilde bir sınıf aynı isimde birden fazla metot içerebilmektedir. Metotların ayrıştırılmaları derleme zamanında yapılmaktadır. Oluşturulan tüm bu metotlar birbirlerine benzer işleri yerine getirmektedirler, fakat işlemlerini farklı şekillerde yapmaktadırlar. Bu nedenle tüm bu metotlara uyum hesaplama sürecinde farklı metotlar olarak davranılmalıdır.

Metot ezme, temel tip içinde yazılan bir metodun aynı imza ile kalıtılan tip içerisinde de yer alabilmesine izin vermektedir [51]. Kalıtılan tip içerisinde yazılan metot aynı isim ve imzaya sahip olduğu temel tipte yer alan metodun yerini alabilmektedir. Bu ayrıştırma çalışma zamanında yapılmaktadır.

Bir nesne diğer bir nesne ile haberleşebilmesi için, nesnenin diğer nesneye mesaj aktarması gerekmektedir [52]. Mesaj aktarma, basit bir ifade ile bir nesnenin durumuna erişebilmek ve nesnenin durumunu değiştirebilmek anlamına gelmektedir. Mesaj aktarma diğer bir ilişki olan, etkileşim ilişkisini ifade etmektedir. Etkileşim

ilişkisi temel olarak metotlar için tanımlıdır. Uyum ölçütlerinde temel olarak kullanılan karakteristiklerden biri de etkileşimdir.

Yukarıda bahsedilen sınıf karakteristiklerine bakıldığında metotlar ve sınıflar uyum için temel hesaplama bloklarını ifade etmektedirler. Çünkü yapısal anlamda uyum yukarıda bahsedilen etkileşimler yolu ile oluşmaktadır. Etkileşimlerin kaynağı ise metotlardır. Sınıflar ise sarmalama kuralının gereği etkileşimleri tutan yapıyı teşkil etmektedir. Bu açıdan etkileşimlerin kaynağını oluşturan metotları içeren ve uyumu etkileyen nesneye yönelik değerler dizisi içinde en küçük ve temel blok görevindedir.

Tüm aktarılan bu karakteristikler, nesneye yönelik sistemlerin ve sınıfların karakteristikleridir. Eğer bu tür sistemler için bir uyum ölçütü hazırlanacaksa kesin olarak bu karakteristiklere ve özelliklerine dikkat edecek şekilde geliştirilmelidir. Hazırlanacak ölçütler bu özellikleri karşıladığı kadarı ile verimli ve etkin olacaklardır. Bu özellikler karşılanmadığı sürece sadece belirli basit sayılar üzerinden yapılan hesaplar kaba bir sonuca sebep olacaktır ve etkinliği her zaman soru işareti olarak kalacaktır.

2.6. Tek Sorumluluk Prensibi

Sınıf karakteristikleri yanında nesneye yönelik tasarım alanında önemli özelliklerden birisi tek sorumluluk prensibidir (Single Responsibility Principle SRP). Bir sistem nesneye yönelik tasarım kuralları ile tasarlanmak istendiğinde göz önünde tutulması gereken özelliklerden birisi olan tek sorumluluk prensibi aynı zamanda sınıf uyumu içinde önemli bir yer tutmaktadır. Tek sorumluluk prensibine göre bir sınıf sadece tek bir görevi yerine getirmelidir [53, 54]. Tom Demarco ve Meilir Page-Jones bu prensibi doğrudan uyum olarak adlandırmışlardır. Bir sınıfın değiştirilmesi için bir sebepten fazlası olmamalıdır.

Sınıf içindeki tüm üyeler bir amacı yerine getirmede bir görev sahibi olmalıdırlar. Tüm bu görevlerin bütünü ise sonuç olarak amaca hizmet etmeli ve sonucu belirlenen bu amaç olmalıdır. Eğer bir şekilde bir sınıf birden fazla amaç için bir araya gelmiş üyelerden oluşuyorsa, kodun karmaşıklığı artmakta, bakımı zorlaşmakta

ve değişiklikler sistemi daha zorlayıcı hale getirmektedir. Bu durumda rahatlıkla sınıfın kötü bir uyuma sahip olduğunu söyleyebilmekteyiz.

Farklı sorumluluklar tek bir sınıfa atandıkları zaman sınıf içinde birbirinden bağımsız bileşenler grubu oluşmaktadır. Bu durumda eğer bir sınıf birden fazla sorumluluğa sahip ise sınıfın değişmesini tetikleyecek birden fazla sebep oluşacaktır. Bir sınıf birden fazla sorumluluğa sahip olduğunda sorumluluklar bağımlı hale gelmiş olurlar. Bir sorumluluk için oluşan bir etki diğer sorumluluğun yerine getirilmesini engelleyecektir. Bu tür bir tasarım kırılgan nitelik taşımaktadır ve bir değişim oluştuktan sonra önceden tahmin edilemeyecek kırılmalara sebep olacaktır. Bu kapsamda bahsedilen sorumluluk, değişim için bir sebebi ifade etmektedir. Eğer bir sınıfın değişmesi için birden fazla sebep var ise sınıfın birden fazla sorumluluğu vardır denilebilir. Bazı durumlarda bunu görmek oldukça zordur. Uygun şekilde tasarlanmış bir uyum ölçütü kesinlikle bu durumu uyum hesaplama sürecine katmalıdır. Aksi halde uyum beklendiğinden fazla olarak elde edilecektir.

SRP, en basit ve layıkıyla yerine getirilmesi en zor olan prensiplerden biridir. Sorumlulukların tespit edilmesi ve ayrılması gerçekte yapılan tasarım ile alakalıdır. Bu kapsamda önerilen ölçüt bu prensibe önem verdiği için sonuçları sınıfın içerisinde ayrık grup oluşup oluşmadığına göre şekillendirmektedir. Eğer sınıf içerisinde birbirinden bağımsız bileşenler grubu oluşursa uyumu olabileceğinden düşük olarak hesaplayarak ilgili kişilerin dikkatini çekebilmektedir. Ayrıca önerilen ölçüt sadece bir uyum değeri elde etmenin yanında ilişkileri de açığa çıkardığından bilgilendirici bir yapıya sahiptir.

2.7. Yazılımın Kalitesi

Yazılım kalitesi, bazıları tarafından gereksinimlere tam uygunluk olarak tanımlanır. Bazıları ise yazılımın uygulamalı görevlerin etkin bir şekilde ve memnun edici şekilde yerine getirmesidir. Elektrik Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers IEEE)'nün Yazılım Mühendisliği Terminolojisi Standart Sözlüğü' ne göre yazılım kalite disiplini, yazılımın inşaası esnasında kaliteyi temin eden, yapılan planlı ve sistematik aktiviteler kümesi olarak ifade

edilmektedir [55]. Bu tanıma göre yazılım kalitesi yazılım kalite güvencesi, yazılım kalite kontrolü ve yazılım kalite mühendisliğini içermektedir. IEEE sözlüğü yazılım kalitesini (1) bir sistemin, bileşenin veya sürecin belirlenen gereksinimleri sağlama derecesi, (2) bir sistemin, bileşenin ya da sürecin müşteri ya da kullanıcı ihtiyaç ve beklentilerini karşılama derecesi olarak tanımlamaktadır. Bu tanımlardan sonuç olarak yazılım kalitesi şu şekilde tanımlanabilir: Yazılım kalitesi, belirlenmiş ihtiyaç duyulan ihtiyaçları yerine getirecek bir yazılım sisteminin bir küme karakteristiğini ifade etmektedir. Belirlenen bu ihtiyaçlar karakteristikler ve alt karakteristikler olarak tanımlanır ve bu karakteristikler "ölçütler" yardımı ile değerlendirilirler. Belirlenen bu ihtiyaçlar ölçütlerin değerleri bazında ifade edilirler.

Yazılım kalitesi kapsamında iyi kalite; yazılım sisteminin, gereksinimlerin çoğunu ya da tümünü yerine getirmesidir. Kötü kalite; yazılım sisteminin, gereksinimlerin bir kısmını ya da hiçbirini yerine getirememesidir. Ölçütler ise, yazılımın sistemin gereksinimlerini yerine getirip getirmediğini ifade eden göstergelerdir.

2.8. Yazılımın Ölçütü

Yazılım ölçütü, yazılımın veya yazılımın tanımlamalarının bir parçasının bazı özelliklerinin ölçüsüdür. Amacı, kalite güvence ve diğer yazılım geliştirme süreçlerine girdi oluşturacak nicel verileri sağlamaktır. Çünkü nicel veri, projeleri kontrol etmek için önemlidir. Tom DeMarco'nun ifadesine göre ölçemediğinizi kontrol edemezsiniz [56]. Yazılım ölçütleri hakkında teori, özellikle nesneye yönelik ölçütler için [57] içinde bulunabilir. Bu kaynak yazılım ölçütleri tanımlarını özetlemektedir. Buna göre yazılım ölçütleri yazılım projesinin belirli bazı özelliklerini ölçen ve bu ölçümleri iyi tanımlı objektif ölçüm kurallarına göre yapan yardımcılardır.

2.9. Ölçme ve Yazılım Ölçütü Kavramları

Bir yazılım ölçütü, diğer disiplinlerdeki ölçümlerde olduğu gibi, genel geçer kabul görmüş ölçüm oluşturma kurallarına uygun olmalıdır [58]. Bunun yanında, yeni bir ölçüt geliştirilirken basit fakat temel prensipler uygulanmalıdır. Ölçüm, gerçek

dünyadaki varlıkların özelliklerinin, belirlenmiş iyi tanımlı kurallar yardımı ile bazı sayılar ve semboller vasıtası ile ifade edilmesidir. Bir varlık, bir yazılım parçası veya insan gibi, herhangi bir şey olabilir. Varlıkların özellikleri, varlığın sahip olduğu özelliklerdir, örneğin bir yazılım için kodun boyutu, bir insan için boy uzunluğu. Tüm bunlar bazı yönlerden ölçülebilirdir.

Ölçümün doğası gereği, bir özellik ölçülürken sayılar belirli bir şekilde sezgisel ya da deneysel gözlem ile varlığın özelliğine atanırlar, bu noktada özelliğin diğerlerine karşı olan durumları korunur. Örneğin, insanların boyları ölçülürken, uzun insanlara daha büyük rakamlar verilir.

Bazen bazı özelliklerin kişiden kişiye değişen sezgisel anlamları bulunmaktadır. Bu sezgisel anlam değerlendirilecek özelliğin modelini elde etmede önemlidir. Zira bir model belirli bir bakış açısını ifade eder. Örneğin insan için boy ölçümünde ayakkabının olup olmadığına göre boy ölçüsünün değiştiği gibi. Bu durum özellikle yazılım ölçümlerinde iyi modeller oluşturma da gereklidir, çünkü yazılım özellikleri değerlendirilme açısından karmaşık yapıya sahiptirler ve ölçümde bakış açıları oluşturmak önemlidir. Bu açıdan günümüze kadar herkes tarafından kabul gören ölçütlerin olmayışı bu noktayı açıklamaktadır.

Ölçme ayrıca, doğrudan ya da dolaylı olarak iki şekilde yapılabilmektedir. Bir şeyin ölçülmesi doğrudan ölçülen şeyin değerlendirilmesi şeklinde ise doğrudan ölçme yapılıyor demektir. Dolaylı ölçümde ise bir şeyin ölçümü bazı diğer özelliklerin ölçümü yardımı ile yapılır. Bu açıdan bakıldığında sınıflar için uyum ölçütü geliştirmek bir anlamda sınıfın üyeleri arasındaki ilişkileri ölçmek gibi doğrudan yapılan ölçümleri gerektirecektir. Sınıf uyumu bu durumda dolaylı ölçüme girecektir.

Ölçüt geliştirmenin iki aşaması vardır. Öncelikle, ölçüt öncelikle tanımlanmalı ve diğer ölçütlerden farkları ortaya konulmalıdır. İkinci olarak ölçütü doğrulayacak deneysel çalışmaların yapılması gerekmektedir. Yazılım açısından yazılım ölçütü tanımlamayı Fenton şu adımlarla açıklamıştır [58]: Gerçek dünya varlığı için ölçülecek özelliğin tespiti, bu varlık için deneysel ilişkilerin belirlenmesi, her bir deneysel ilişki için sayısal değerlerin atanması, varlık ile sayıların eşleştirilmesi ve

verilen sayılar ile deneysel ilişki arasındaki bağın korunup korunmadığının kontrol edilmesidir.

Bu bölüm içerisinde nesneye yönelik programlama dillerine ait temel karakteristiklerden bahsedilmiş ve önerilen ölçüt açısından önemli görülen özellikler üzerinde durulmuştur. Bunların yanında yazılım kalitesi ve yazılımı ölçütü hakkında temel bilgiler verilmiştir. Takip eden bölüm içerisinde ise mevcut kabul görmüş metrikler detaylı olarak aktarılacaktır.

3. MEVCUT ÖLÇÜTLER

3.1. Giriş

Bu kısım, giriş kısmında kısmen verilen nesneye yönelik uyum ölçütlerinin detaylı olarak aktarılmasını içermektedir. Literatürde günümüze kadar uyum ölçütleri ile yapılan çok sayıda çalışma olmasına karşın genel geçer uyum ölçütü kabulü bulunmamaktadır. Bu durumun sebepleri bir önceki bölümde aktarılmıştır. Fakat bazı ölçütler tarih içerisinde diğerleri için yönlendirici ve esin kaynağı olduğu için kabul görmüşlerdir. Kabul görmüş bu ölçütlerin detaylı bir şekilde incelenmesi oluşturulacak yeni ölçütler için büyük önem taşımaktadır. Bu bölüm tez kapsamında sunulacak olan ölçüte yönlendirici nitelikte bulunan ölçütlerin detaylı açıklamalarını kapsamaktadır.

3.2. Uyum Ölçütleri

3.2.1. LCOM Genel

Kabul edilmiş ölçütlere bakıldığında LCOM ölçütünün çeşitli versiyonlarına rastlanmaktadır. LCOM nesneye yönelik uyum ölçütlerinin başlangıcı olarak kabul görmektedir. Bu ölçüt literatürde geniş olarak kullanılmıştır. LCOM temel olarak uyum eksikliğine odaklanmaktadır. Metot benzerliğini ortak paylaşılan öznitelikler üzerinden belirleyerek, benzer olmayan metot çifti sayısını temel almaktadır. Benzer olmayan metot sayılarını işleyerek uyum değerini elde etmektedir. Bu temel ölçütte üyeler arasında yer alan diğer etkileşimler işlenmemektedir. Diğer yaklaşımlar bu benzerlik prensibini kullanarak kendi eklentileri ile yeni ölçütler sunmuşlar ya da kendi ölçütlerini geliştirmişlerdir. Bu ölçütler detaylı olarak [5] içinde verilmektedir.

Tablo 3.1’de mevcut ölçütlerin listesi verilmektedir.

Tablo 3.1: Kabul görmüş ölçütler

LCOM1	Ortak nesne değişkeni kullanmayan metot çiftlerinin sayısıdır [14].
LCOM2	P, nesne değişkenleri paylaşmayan metot çiftlerinin sayısı ve Q nesne değişkenleri paylaşan metot çiftlerinin sayısı ise bu durumda eğer |P|>|Q| ise LCOM2=|P|-|Q|, aksi halde LCOM2=0 olur [46].
LCOM3	Köşeleri metotlar, metotlar en azından bir nesne değişkeni paylaşıyor ise aralarında kenar olan yönlendirilmemiş bir G grafiği düşünüldüğünde, LCOM3=|G'nin bağlantılı bileşenlerinin sayısı| [3].
LCOM4	LCOM3'e benzer, G grafiğine ek olarak, Mi metodu, Mj metodunu işletiyorsa metotlar arasına kenarlar çizilir [17].
Co	LCOM4'teki G grafiğinin köşeleri V, kenarları E olsun. Bu durumda $Co = 2 . \frac{\mid E \mid - (\mid V \mid - 1)}{(\mid V \mid - 1) . (\mid V \mid - 2)}$ [17].
LCOM5	Mi (i=1,..,m) metot kümesinin Aj (j=1,..,a) nesne değişkeni kümesine eriştiğini düşünelim. μ(Aj), Aj nesne değişkenini kullanan metot sayısı olsun. $LCOM5 = \frac{\frac{1}{a}\left(\sum_{j=1}^{a} \mu(Aj)\right) - m}{1 - m}$ [15].
Coh	LCOM5'in bir çeşididir. $Coh = \frac{\sum_{j=1} \mu(Aj)}{m \cdot a}$ [5].
TCC	N adet metoda sahip bir sınıf düşünüldüğünde, NP, azami metot çiftlerinin sayısını vermektedir: NP=[N*(N–1)]/2. NDC, metotlar arasında doğrudan bağlantıların sayısı olursa, TCC, doğrudan bağlı yerel metotların göreli sayısını verir: TCC = NDC / NP [4].
LCC	NIC metotlar arasında doğrudan ya da dolaylı bağlantıların sayısı olursa, LCC, doğrudan ya da dolaylı bağlı metotların göreli sayısını verir: LCC=NIC/NP [4].

3.2.2. LCOM1

Nesneye yönelik ölçütler içinde en çok örnek alınan ölçüt Chidamber ve Kemerer tarafından önerilen ölçüttür [1]. Metotlardaki uyumun eksikliğini ifade eden bu ölçüt LCOM (Lack of Cohesion in Methods) olarak adlandırılmıştır. Bu ölçütün tanımı şu şekildedir:

M_1, M_2, … ,M_n metotlarını içeren bir C sınıfının olduğunu düşünelim. M_i metodu tarafından kullanılan öznitelik kümesi I_i olsun. Bu durumda n tane metot tarafından kullanılan öznitelikleri içeren n tane küme oluşacaktır, $\{I_1\}$-$\{I_n\}$. LCOM1, n tane kümenin kesişimleri ile oluşan ayrık kümelerin sayısıdır. Diğer bir tanım ise, ortak öznitelik kullanmayan metot çiftlilerinin sayısı olarak karşımıza çıkmaktadır. LCOM1 temel olarak metotların benzerliğini kullanmaktadır. Metot benzerliği ise ortak paylaşılan öznitelik ile sağlanmaktadır. Metotlar arasındaki etkileşimler bu ölçüt kapsamında hiç değerlendirilmemiştir. LCOM1 tarafından üretilen uyum değerleri belirli bir aralık içinde değildir. Bu açıdan normalize edilmiş değillerdir. Sınıf içindeki etkileşimin artması bazı durumlarda uyum değerini değiştirememektedir.

LCOM1 aşağıda da anlatılacağı gibi literatürde fazlaca değerlendirilmiş ve farklı yorumları sunulmuştur.

3.2.3. LCOM2

Chidamber ve Kemerer daha sonraki çalışmasında LCOM'u (LCOM2) tekrar yorumlamışlardır [46]:

M_1, M_2, … ,M_n metotlarını içeren bir C sınıfının olduğunu düşünelim. M_i metodu tarafından kullanılan öznitelik kümesi I_i olsun. Bu durumda n tane metot tarafından kullanılan öznitelikleri içeren n tane küme oluşacaktır, $\{I_1\}$-$\{I_n\}$. P, nesne değişkenleri paylaşmayan metot çiftlerinin sayısı ve Q nesne değişkenleri paylaşan metot çiftlerinin sayısı olsun. $P = \{(I_i, I_j) \mid I_i \cap I_j = \emptyset\}$ ve $Q = \{(I_i, I_j) \mid I_i \cap I_j \neq \emptyset\}$. Bu durumda $LCOM = |P| - |Q|$ şeklinde tanımlanmaktadır. Eğer tüm öznitelik kümeleri boş küme ya da $|P| <= |Q|$ ise uyum değeri 0 olarak kabul edilmektedir.

Bu versiyon da normalize edilmiş değildir. Sınıf içinde benzer metotların birden fazla öznitelik paylaşmaları uyumu değiştirebilir bir durum oluşturamamaktadır. Bunun gibi birçok durumda uyumu doğru bir şekilde elde edememektedir. Diğer sürümler bu eksiklikleri gidermeye çalışmışlardır.

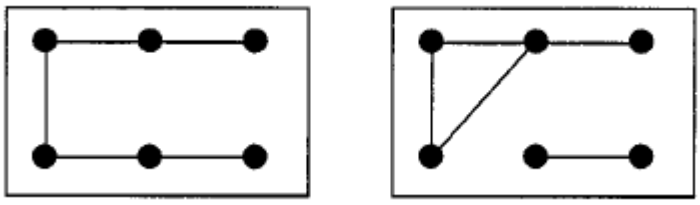
3.2.4. LCOM3

LCOM ölçütü Li ve Henry tarafından farklı bir şekilde tanımlanmıştır [3]. Buna göre metotların ayrık kümelerinin sayısıdır. Eğer herhangi iki metot ortak bir öznitelik kullanıyorsa, aynı kümede yer almaktadırlar. Bu değer 0 ile N arasındadır. N değeri en fazla metot sayısı kadar olabilmektedir ve pozitif bir değerdir.

Hitz ve Montazeri, Li ve Henry'nin tanımını grafiksel bir tanım içerecek şekilde sunmuşlardır [60]. Sunulan ölçüt şu şekilde tanımlanmıştır:

X bir sınıfı, I_X X sınıfının öznitelik kümesini ve M_X X sınıfının metot kümesini gösterdiğini düşünelim. Basit yönlendirilmemiş bir grafik, $G_X(V,E)$; $V=M_X$ ve $E=\{<m,n> Є VxV \mid \square iЄI_X : (m \text{ erişir } i) \square (n \text{ erişir } i)\}$, olarak ifade edilmektedir. Düğümleri ifade eden metotlar arasında eğer metotlar en azından ortak bir öznitelik kullanıyorlarsa kenar çizilmektedir. Buna göre LCOM3 G_X grafiğindeki bağlı bileşenlerin sayısı olarak tanımlanmaktadır. Farklı bir tanımla, özniteliklerin ayrık kümeleri üzerinde işleyen metot demetlerinin sayısıdır. Şekil 3.1, oluşturulan G_X grafiğine dair örnek içermektedir.

LCOM3, ayrık metot gruplarını ortaya çıkarma hususunda oldukça etkilidir. Fakat bir şekilde benzer olan metotlar arasında paylaşılan öznitelik sayısı artsa da üretilen uyum değeri değişmemektedir. Başka bir deyişle üyeler arasında bir şekilde paylaşım oluştuğunda uyum azami değeri almaktadır. Fakat bu ölçüt üyeler arasındaki etkileşim desenine yeteri kadar önem vermemektedir.



Şekil 3.1: G_X grafiğine dair örnek

3.2.5. LCOM4

Hitz ve Montazeri ayrıca sundukları ölçütün ikinci bir versiyonunu (LCOM4) da önermişlerdir [17]. LCOM4, LCOM3'e benzemektedir ve G_X grafiğine ek olarak eğer M_i metodu M_j metoduna metot çağrımı ile erişiyorsa ise bu metotlar arasında kenar çizilmektedir. Bu tür eklentinin sebebi erişim metotları adı verilen özel metotların sınıf içerisinde sadece bazı özniteliklere erişimi sağlamasıdır. Eğer bu tür metotlar uyum sürecinde doğrudan bırakılırsa diğer normal metotlarla hiçbir zaman benzerlik göstermeyecektir. Bu durumu giderebilmek için öznitelik kullanımı yanında metotların birbirlerini çağırmaları da grafiğe eklenmektedir. Bir şekilde metotların özniteliklere özel metotlar üzerinden erişebileceği olasılığı üzerinde durulmaktadır.

LCOM4 de LCOM3'ün sahip olduğu dezavantajlara sahiptir. Uyum elde etme sürecinde, metotlar arasında bir şekilde benzerlik yakalandığında uyum değeri azami olacak şekilde değerlendirilmektedir.

3.2.6. Co

Hitz ve Montazeri LCOM4'e ek olarak C olarak isimlendirilen bir ölçüt sunmuşlardır [17]. Briand'ın [5]'deki çalışmasında bu ölçütten Co (Connectivity) olarak bahsedilmiştir. Co sadece tek bir bağlı metot demeti olduğunda uygulanmaktadır. Co bağlı metot demeti içindeki kenarları saymaktadır ve hesap sonucunu 0 ve 1 arasına denk gelecek şekilde düzenlemektedir. Co şu şekilde ifade edilmektedir (E:Kenar, V:Köşe):

$$Co = 2 . \frac{|E| - (|V| - 1)}{(|V| - 1) . (|V| - 2)}$$

Bu durumda daima Co(c) $\in$ [0, 1] durumu sağlanacaktır. 0 ve 1 değerleri sırası ile $|E_C|=|V_C|-1$ ve $|E_C|=|V_C|(|V_C|-1)/2$ değerlerinden hesaplanmaktadır. Bu ölçüt

normalize edilmiştir. Fakat yine de bir şekilde benzer olan metotlar arasındaki etkileşim desenini işleyememektedir.

3.2.7. LCOM5

Henderson-Sellers LCOM ölçütünü benzer şekilde tanımlamışlardır (LCOM5) [15]. Bir küme metodun {Mi} (i=1, … , m) bir küme özniteliğine {Aj} (j=1, … , a) eriştiğini düşünelim. Her bir Aj özniteliğini kullanan metot sayısı μ(Aj) olarak kabul edildiğinde LCOM5 şu şekilde ifade edilmektedir:

$$\text{LCOM5} = \frac{\frac{1}{a}\left(\sum_{j=1}^{a} \mu(\text{Aj})\right) - \text{m}}{1 - \text{m}}$$

Henderson-Sellers yukarıda bahsi geçen ifade için aşağıda belirtilen ifadeleri kullanmışlardır:

- Ölçüm, eğer her bir metot her özniteliğe erişiyor ise 0 değerini alır ve bu durum mükemmel uyum olarak adlandırılır.
- Ölçüm, eğer her bir metot sadece bir tek özniteliğe erişiyor ise 1 değerini alır.
- 0 ve 1 arasında yer alan değerler mükemmel değerin oranlarını ifade etmektedir.

LCOM5 diğer versiyonlara nazaran, hesaplama sürecine doğrudan metotlar tarafından kullanılan öznitelik sayılarını da katmaktadır. Bu şekilde metotların eriştikleri öznitelik sayısı uyumu değiştirmektedir. Oluşturulan bu durum uyum hususunda daha hassas sonuçların elde edilebilmesine imkân sağlamaktadır.

3.2.8. Coh

Briand et al. LCOM5'in farklı bir versiyonunu tanımlamışlardır [5]. Bu ölçüt Coh (Cohesion) olarak isimlendirilmektedir. Bazı öznitelikler hiçbir metot tarafından

erişilmiyor olabilmektedir. Orijinal ölçüt bunu dikkate almamaktadır. Coh şu şekilde ifade edilmektedir:

$$\text{Coh} = \frac{\sum_{j=1} \mu(\text{Aj})}{\text{m} \cdot \text{a}}$$

3.2.9. TCC ve LCC

Bieman ve Kang daha farklı bir uyum ölçütü tanımlamışlardır [4]. Ortak öznitelik paylaşan metot çiftlilerini kullandığından dolayı bu ölçüt LCOM ile benzer özellikler taşımaktadır. Bununla birlikte, öznitelik kullanımını farklı bir şekilde hesaplamaktadırlar. Metotların öznitelik kullanımını doğrudan ve dolaylı olacak şekilde farklı tanımlamışlardır. Bir m metodu, eğer doğrudan bir özniteliği okuyor ya da özniteliği değiştiriyor ise, metodun özniteliğe doğrudan eriştiği kabul edilir. Eğer bir M_i metodu metot çağrımı ile bir M_j metoduna erişiyor ve M_j metodu bir özniteliğe doğrudan erişiyor ise m metodunun bu özniteliğe dolaylı eriştiği kabul edilir. Ayrıca iki metot, dolaylı ya da doğrudan ortak bir özniteliğe erişiyor ise bu iki metodun bağlı olduğu kabul edilir. Bieman ve Kang bu kavram üzerine 2 ölçüt tanımlamışlardır: TCC ve LCC.

3.2.9.1. TCC

TCC (Tight Class Cohesion), doğrudan ya da dolaylı olarak ortak özniteliğe erişen bağlı metot çiftlilerinin oranıdır. N adet metoda sahip bir sınıf düşünüldüğünde, NP, azami metot çiftlilerinin sayısını vermektedir: NP=[N*(N–1)]/2. NDC, metotlar arasındaki doğrudan bağlantıların sayısı olursa, TCC= NDC/NP [4].

3.2.9.2. LCC

LCC (Loose Class Cohesion), TCC ölçütüne benzemektedir, ortak özellik kullanan metotlar için geçişlilik özelliğinide katmaktadır. Başka bir ifade ile ortak özniteliğe doğrudan ya da dolaylı olarak erişen metot çiftlilerini esas almaktadır. Buna göre

NIC, metotlar arasında doğrudan ya da dolaylı bağlantıların sayısı olursa, LCC=NIC/NP [4].

Diğer ölçütlerde bahsedilmeyen bir husus TCC/LCC ölçüt çiftinde ortaya çıkmaktadır: Kalıtım. Bieman ve Kang kalıtımın uygulanması durumunda sınıfın uyumunu hesaplamak için 3 seçenek sunmuşlardır.

- Kalıtılmış metot ve özniteliklerin uyum süreci dışında tutulması.
- Kalıtılmış metot ve özniteliklerin uyum sürecine katılması.
- Kalıtılmış metotların uyum süreci dışında tutulması, kalıtılmış özniteliklerin uyum sürecine katılması.

Kalıtım haricinde Bieman ve Kang yapıcı adı verilen özel metotların uyumu yapay bir şekilde arttırdığını göstermişlerdir. Bu sebeple uyum hesaplama sürecinden yapıcıları çıkarmışlardır. Kalıtım ve yapıcı metotlar üzerine tanımlanan davranışlar tamamen yeterli olmasalar da fikir vermek açısından çok faydalı olmuşlardır.

Bu bölüm kapsamında, kabul görmüş mevcut ölçütler ayrıntılı şekilde aktarılmıştır. Geçen bölümde anlatılan genel karakteristikler ve bu bölüm kapsamında elde edilen bilgiler ışığında diğer bölümde önerilecek olan ölçütün matematiksel modeli çıkarılacak ve modele dair ayrıntılı açıklamalar yapılacaktır.

4. İLİŞKİ TABANLI UYUM (ROC)

4.1. Giriş

Bu bölümde, önerilen yaklaşımın tanımları ve matematiksel modeli adım adım aktarılacaktır. Geliştirilen ölçüt tez kapsamında "İlişki Tabanlı Uyum (Relation Oriented Cohesion)" olarak isimlendirilecektir. Geliştirilen ölçüte bu ismin verilmesi, sınıf karakteristiklerini özellikle sınıf üyeleri arasındaki ilişkileri farklı bir şekilde ve detaylı olarak ele almasından kaynaklanmaktadır. Sınıf için uyumu ortaya çıkaran temelde sınıf üyeleri arasındaki ilişkilerdir. Üyeler arasındaki farklı ilişkileri yakalayabilmek ya da bu ilişkileri uygun şekilde işleyebilmek uyumu uygun şekilde elde etmenin esas unsurudur. Tabii ki bu durum tek başına yeterli değildir. Geliştirilen ölçüt, temelde ilişkileri esas aldığı için "İlişki Tabanlı Uyum" ölçütü olarak isimlendirilmiştir. Bu bölümde ayrıca geliştirilen ölçütün teorik olarak değerlendirilmesi yapılacaktır.

4.2. Ölçüt Tanımları

Tanım 1. Nesneye yönelik bir sistem, bir sınıf kümesinden oluşmaktadır, C. Her bir sınıf c Є C şeklinde tanımlıdır. c sınıfı bir küme öznitelik A(c), bir küme metot M(c) ve bir küme etkileşim sayılarını I(c) içermektedir.

$$\mathrm{A(c)} = \{a_1, a_2, \ \ldots \ , a_j\}$$

$$\mathrm{M(c)} = \{m_1, m_2, \ \ldots \ , m_i\}$$

$$\mathrm{I(c)} = \{im_1a_1, im_1a_2, \ \ldots \ , im_ia_j\}$$

Metotlar, sınıf fonksiyonelliğinin kaynağıdır. Bu çalışmada, metotlar ayrıca sınıf uyumunun kaynağı olarak kabul edilmektedir. Uyumlu metotlar sınıfı daha uyumlu, uyumsuz metotlar sınıfı uyumsuz hale getirirler. Metot uyumu ise doğrudan ya da dolaylı olarak özniteliklere erişilmesi olarak tanımlanmaktadır [32]. Ayrıca sunulan

yaklaşım kapsamında erişim yanında, etkileşim sayıları ve etkileşim tipleri de metot uyumunun elde edilmesinde kullanılmaktadır.

Tanım 2. Bir m_i metodu için, $A_D(m_i)$, m_i tarafından doğrudan erişilen öznitelik kümesidir ve $A_I(m_i)$, m_i tarafından metot işletimleri vasıtası ile dolaylı olarak erişilen öznitelik kümesidir.

Bir m_i metodu için, $A_{DW}(m_i)$, m_i tarafından doğrudan yazılan öznitelik kümesi, $A_{DR}(m_i)$, m_i tarafından doğrudan okunan öznitelik kümesi ve $A_{IW}(m_i)$, m_i tarafından metot işletimleri vasıtası ile dolaylı olarak yazılan öznitelik kümesidir. Buradan hareketle $A_D(m_i) = A_{DW}(m_i) \cup A_{DR}(m_i)$ $A_I(m_i) \supset A_{IW}(m_i)$ elde edilir.

Metot tarafından doğrudan erişilen öznitelik sayısı artarsa, metodun uyumu artar. Metot tarafından dolaylı olarak erişilen bir öznitelik metot uyumuna katkı yapar, fakat bu katkı göreceli olarak daha azdır [32].

Tanım 3. Bir a_i öznitelik için, $M_D(a_i)$, a_i özniteliğe doğrudan erişen metot kümesidir ve $M_I(a_i)$, a_i özniteliğine dolaylı olarak erişen metot kümesidir.

Bir a_i öznitelik için, $M_{DR}(a_i)$, a_i özniteliğini doğrudan okuyan metot kümesidir ve $M_{IW}(a_i)$, a_i özniteliğine dolaylı olarak yazan metot kümesidir. Buradan hareketle $M_D(a_i) \supset M_{DR}(a_i)$, $M_I(a_i) \supset M_{IR}(a_i)$ elde edilir.

Metot uyumu öznitelik kullanım oranına bağlıdır. Metotlar tarafından çok kullanılan bir öznitelik, onu kullanan metotları daha uyumlu yapar. Bir sınıfın uyumu, üyeler arasındaki etkileşimlerin sayısı ve etkileşim tipi yanında üyeleri arasındaki etkileşim desenine de bağlıdır. Başka bir deyişle, sınıf üyeleri arasındaki bağlılık uyumu etkiler. Öznitelikler ve metotlar uyuma farklı ağırlıklarda katkıda bulunurlar [32].

Tanım 4. Bir a_i öznitelik için, $A_{DA}(a_i)$, a_i özniteliği kullanılarak elde edilen bağımlı özniteliklerin kümesidir.

Bağımlı öznitelikler, değeri doğrudan ya da dolaylı olarak diğer öznitelikler tarafından belirlenen özniteliklerdir. Bu durum söz konusu olduğunda, bağımlı öznitelik diğer özniteliklere akış ya da kontrol bağımlı hale gelmektedir. Ayrıca kendine bağımlı olunan bir özniteliğin kullanımı bağımlı olan özniteliğin kullanımına göre metot uyumuna daha fazla katkı sağlamaktadır.

Bağımlı özniteliklerin bulunabilmesi için, kodun öznitelikleri bulacak şekilde taranması gerekmektedir. Normal atamalarda ya da sonucu bir özniteliğe dönen ve parametre alan fonksiyon çağrımı sonucu doğrudan bir etkileme olduğundan bu tip etkileşimler akış bağımlılığı olarak değerlendirilmektedir. Akış bağımlılığında soldaki öznitelik sağda yer alan özniteliklere bağımlı durumdadır. Eğer kontrol ifadelerinden sonra bir değer ataması söz konusu ise bu etkileşim kontrol bağımlılığı olarak değerlendirilmektedir. Kontrol bağımlılığında, kontrol ifadesi içinde yer alan öznitelikler kontrol özniteliklerine bağımlı hale gelirler. Bu işlemler neticesinde yönlü bir bağımlılık grafiği (Tanım 8) oluşmaktadır.

Tanım 5. Bir m_i metodu ve a_j özniteliği için etkileşim sayısı, $I_{MA}(m_i, a_j)$, m_i' nin a_j' ye kaç defa eriştiğinin sayısı olarak tanımlanır. Erişimin sayısı, özniteliği okuma ve özniteliğe yazma sayılarının toplamı olarak ifade edilmektedir.

Metot uyumu, özniteliklere erişimin yanında özniteliklerle olan etkileşimin sayısına da bağlıdır. Etkileşim sayısındaki artış metot uyumunu göreceli olarak arttırmaktadır. Bir metot özniteliklerle ne kadar fazla etkileşim içinde ise uyumu o oranda artacaktır. Bu durum etkileşim sayısının bir olduğu duruma göre sınıf uyumunu da belirli oranda arttırmakta ve daha hassas değerlerin elde edilmesine yardımcı olmaktadır. Bu artış sınıf üyeleri arasındaki erişimlerin tam olduğu duruma kadar devam edecektir. Bu durumda sınıf uyumu artmayacak fakat metotların uyumları etkileşim sayılarına göre farklılık gösterecektir ve sınıf uyumuna olan katkıları değişecektir.

Ayrıca üyeler arasındaki etkileşimin tipi özellikle yazma olduğunda metot uyumuna katkısı artmaktadır. Bunun sebebi, yazılan özniteliğin onu kullanan diğer metotları etkilemesinden kaynaklanmaktadır. Bunun yanında uyum değeri erişim sayısı sebebi ile aynı olan metotların durumu, etkileşim sayılarının ve tiplerinin sürece katılması

ile farklılık göstermekte, böylece metot uyumu üzerinden yapılan analizler için daha uygun bir ortam oluşmaktadır. Bu durum uyum yanında metotlar için düzeltici faaliyet gerektiğini ilgili kişilere bildirmek içinde faydalıdır. Örneğin 4 metoda sahip tam bağlantılı bir sınıf için metot uyumları; eğer sadece metodun özniteliklere erişimine bağlı ise; birbirlerine eşit olacaklardır. Ama sınıf karakteristiklerini temel alan ve etkileşimleri çeşitlendiren bir yapıda, metot uyumları birbirinden farklı olması ihtimali yüksektir. Bu durumda ilgili kişiler metot uyumlarına bakarak, yüksek uyuma sahip bir sınıf içindeki dengesiz metot uyumlarını yakalayabilirler. Önerdiğimiz yaklaşımın farklı bir faydası da bu şekilde ortaya çıkmaktadır.

Tanım 6. Bir m_i metodu için, $M_{DC}(m_i)$, m_i tarafından doğrudan çağrılan metot kümesidir ve $M_{IC}(m_i)$, m_i tarafından metot işletimleri vasıtası ile dolaylı olarak çağrılan metot kümesidir.

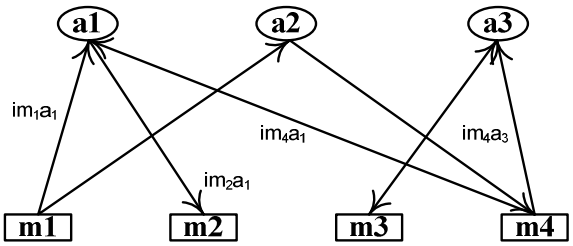
Tanım 7. Bir c sınıfı için, metot-öznitelik kullanım grafiği $G_{MA}(c)$, c'nin üyeleri arasındaki etkileşimin bir gösterimidir ve $G_{MA}(c) = (V, E, I)$ şeklinde yönlendirilmiş bir grafik olarak tanımlanır. Tanımda yer alan V (Vertice [Köşe]) köşe noktalarını ifade etmektedir. Köşe noktalarını metotlar ve öznitelikler oluşturmaktadır. E (Edge [Kenar]) kenarları ifade etmektedir ve kenarlar metotlar ile öznitelikler arasındaki kullanım ilişkilerini göstermektedir. I (Interaction [Etkileşim]) etkileşimleri ifade etmektedir ve kenarların niteliklerini göstermektedir. Metotlar ile öznitelikler arasındaki erişim sayılarını ifade etmektedir.

$$V = A(c) \cup M(c)$$

$$E = \{(m,a) \mid a \in A_D(m), m \in M(c), a \in A(c)\}$$

$$I = \{ima \mid a \in A_D(m), m \in M(c), a \in A(c)\}$$

$G_{MA}(c)$ grafiği üzerinde;
Eğer $m \in M(c)$ ve $a \in A_{DR}(m_i)$ ise ilişki $m \leftarrow a$ şeklinde,
Eğer $m \in M(c)$ ve $a \in A_{DW}(m_i)$ ise ilişki $m \rightarrow a$ şeklinde,
Eğer $m_1, m_2 \in M(c)$ ve $m_2 \in M_{DC}(m_1)$ ise ilişki $m_1 \text{---} > m_2$ şeklinde gösterilmektedir.



Şekil 4.1: Metot-nesne değişkeni kullanım grafiği

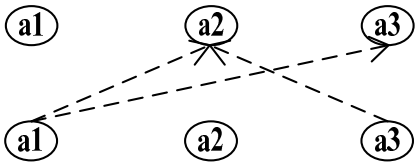
Örneğin, Şekil 4.1 metot-öznitelik kullanım grafiğini göstermektedir. Grafik m_1, m_2, m_3 ve m_4 metotları ve a_1, a_2 ve a_3 öznitelikleri arasındaki etkileşimleri yani erişimleri, etkileşim desenini ve etkileşim sayılarını yönlü olarak göstermektedir.

Tanım 8. Bir c sınıfı için, öznitelik bağımlılık grafiği $G_{AA}(c)$, c'nin öznitelikleri arasındaki etkileşimin bir gösterimidir ve $G_{AA}(c) = (V, E)$ şeklinde yönlendirilmiş bir grafik olarak tanımlanır. Tanımda yer alan V (Vertice [Köşe]) köşe noktalarını ifade etmektedir. Köşe noktalarını öznitelikler oluşturmaktadır. E (Edge [Kenar]) kenarları ifade etmektedir ve kenarlar öznitelikler arasındaki kullanım ilişkilerini göstermektedir.

$$V = A(c) \cup A(c)$$

$$E = \{(a_1, a_2) \mid a_1, a_2 \in A(c), a_1 \in A_{DA}(a_2)\}$$

Eğer $a_1, a_2 \in A(c)$ ve $a_1 \in A_{DA}(a_2)$ ise ilişki $a_1 \text{---} > a_2$ şeklinde gösterilmektedir.



Şekil 4.2: Öznitelik bağımlılık grafiği

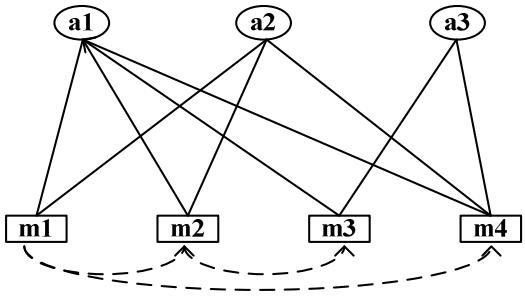
Şekil 4.2, bir sınıfın öznitelik bağımlılık grafiğini göstermektedir. Grafik a_1, a_2 ve a_3 öznitelikleri arasındaki bağımlılık ilişkilerini göstermektedir. Grafiğe göre a_1, a_2 ve/veya a_3 kullanılarak elde edilmektedir. a_3 ise a_2 kullanılarak elde edilmektedir. Bu tür bağımlılıklar kod içerisinde taranır ve grafik üzerinde gösterilirler. Bağımlılığın türü veya sayısı hakkında herhangi bir işlem yapılmaz.

Tanım 9. m_i ve a_j için dolaylı erişim metot işletim yolu, $M_{IP}(m_i, a_j)$, m_i'den a_j'ye erişmek için m_i tarafından doğrudan ya da dolaylı olarak çağrılan metot kümesi şeklinde tanımlanır. $M_{IP}(m_i, a_j)$ içinde birden fazla yol bulunabilmektedir ve içindeki her bir yol bir $IP(m_i, a_j)$ şeklinde tanımlanmaktadır. IP'lerin içerisindeki en kısa yol ise $M_{SP}(m_i, a_j)$ ile tanımlanmaktadır.

$$\mathrm{M_{IP}}(\mathrm{m_i}, \mathrm{a_j}) = \left\{ \begin{array}{l} (\mathrm{m_i}, n_1, n_2, .., n_k) \\ \left(\begin{array}{l} (a_j \in A_I(m_i) \wedge n_1 \in M_{DC}(m_i)) \wedge \\ (a_j \in A_I(n_1) \wedge n_2 \in M_{DC}(n_1)) \wedge ... \\ (a_j \in A_I(n_{k-1}) \wedge n_k \in M_{DC}(n_{k-1})) \wedge \\ (a_j \in A_D(n_k)) \end{array} \right) \\ , m_i, n_1, .., n_k \in M(c) \end{array} \right\} \tag{4.1}$$

$$\mathrm{M_{IP}} = \left\{ IP(\mathrm{m_i}, \mathrm{a_j}) \;\middle|\; m_i \in M(c), a_j \in A_I(m_i) \right\} \tag{4.2}$$

$$\mathrm{M_{SP}}(m_i, a_j) = \left\{ \begin{array}{c} 1,\ \mathrm{a_j} \in A_D(m_i) \\ \min\left(\left| IP(m_i, a_j) \right| \right), \forall IP \in M_{IP}, a_j \in A_I(m_i) \end{array} \right. \tag{4.3}$$



Şekil 4.3: M_{IP} örneği

Örneğin, Şekil 4.3'e göre, m_1 a_3'ü doğrudan kullanmamaktadır. Örneğe göre M_{IP} içinde iki IP bulunmaktadır. Birinci IP (m_1,m_2,m_3) ve ikinci IP (m_1,m_4). Bu durumda, M_{SP} değerine bakılır. M_{SP} değeri en küçük olan seçilir. Örneğe göre, uygun yol (m_1,m_4), çünkü bu yol için M_{SP} değeri 1/2 ve diğeri için 1/3 olarak hesaplanmaktadır. Eğer özniteliğe doğrudan erişim mümkün ise bu durumda M_{SP} değeri bir değerine eşit olmaktadır.

Bir metodun $A_D(m)$ kümesi tek elemanlı küme, $A_I(m)$ kümesi boş küme ve $A_D(m)$ kümesinde yer alan özniteliklerin $M_D(m)$ kümesi boş küme ise, aynı zamanda metot sınıfta yer alan diğer metotlardan en az birinin M_{IP} yolunda yer alıyorsa o metot erişim ya da delegasyon metodu olabilmektedir. Bu durumda, özel metot ölçüm dışında tutulur ve ona yapılan çağrılar doğrudan özniteliğe yapılmış gibi davranılmaktadır. Bunun dışında yapıcı ve yıkıcı gibi sınıf uyumunu yapay olarak etkileyen, literatürde özel metot olarak adlandırılan metotlar, uyum hesaplama süreci dışında tutulurlar. Ayrıca sınıf içindeki metot aşırı yüklemelerinin, her biri ayrı metot olarak değerlendirilirler. Aşırı yüklenmiş metotların farklı metotlar olarak algılanması sınıf karakteristikleri açısından önemlidir. Çünkü yordamsal yaklaşımda benzer işi yapan metotlar ayrı olarak isimlendirilmektedir. Nesneye yönelik sistemler bunun önüne geçerek aynı isim farklı imza ile metot yazabilme imkânı sunmaktadır. Bu açıdan metotları ayrı olarak kabul etmek mantıklı ve gereklidir.

Tanım 10. Bir c sınıfı için, sınıf uyum böleni, $C_{CD}(c)$, sınıfın içinde bulunduğu şartlar altında olabilecek en yüksek uyum değerini göstermektedir.

$$
\begin{aligned}
C_{CD}(c) = & \left[\begin{array}{l} \left(\sum_{m_i \in M(c)} \left| A_D(m_i) \right| \right) + \\ \left(\sum_{m_i \in M(c)} \sum_{a_j \in A_{DW}(m_i)} \left(\left| M_{DR}(a_j) \right| + \left(\sum_{m_k \in M_{IR}(a_j)} 1/M_{SP}(m_k, a_j) \right) \right) \right) \\ \left(\sum_{m_i \in M(c)} \sum_{a_j \in A_{IW}(m_i)} \left| M_{DR}(a_j) \right| / M_{SP}(m_i, a_j) \right) \end{array} \right] * \\
& \left[\sum_{a_j \in A(c)} \left| M_D(a_j) \right| + \sum_{a_j \in A(c)} \left| A_{DA}(a_j) \right| \right] + \\
& \left[\begin{array}{l} \left(\sum_{m_i \in M(c)} \sum_{a_j \in A_D(m_i)} I_{MA}(m_i, a_j) \right) + \\ \left(\sum_{m_i \in M(c)} \sum_{a_j \in A_{DW}(m_i)} \left| M_{DR}(a_j) \right| \right) \end{array} \right]
\end{aligned}
\qquad (4.4)
$$

Sınıf uyum böleni; üyeler arasındaki erişimler ve etkileşimler açısından, sınıfın içinde bulunduğu şartlar altında, olabilecek en yüksek uyum değeri göstermektedir. Bu değer metot uyum değerlerinin belirlenmesinde yardımcı olmaktadır. Fakat azami değer sadece tüm erişimlerin olduğu bir sınıf için erişilebilirdir. Bunun haricindeki durumlarda oransal olarak metotların bir bütün olarak ulaşmaya çalıştıkları uyumu ifade etmektedir. Sonuç olarak metotların uyumu bir bütün olarak bu değere erişmeye çalışmaktadır, etkileşim deseninde olan bir eksiklik ise bu değere erişimi kısıtlamakta ve sınıf uyum değerini hassas olarak elde edilmesine yardımcı olmaktadır.

Sınıf uyum böleni tek bir değer hesabından ziyade çeşitli etkileri bünyesinde barındıran bir değer olarak belirlenmektedir. C_{CD}, metotların doğrudan eriştiği özniteliklerin sayılarını, yazma etkileşim tipi sonucunda etkilenen metotların sayılarını, dolaylı olarak erişilen öznitelik sayılarını, özniteliklerin erişildiği metotların sayılarını, bağımlı değişken etkisini, üyeler arasındaki etkileşimleri içine almakta ve bu değerler arasındaki ilişkileri düzenlemektedir. Sonuç olarak sınıfın içinde bulunduğu şartlar altında en yüksek uyum değeri elde edilmektedir.

Tanım 11. m_i için metot ağırlığı, $WM(m_i)$, m_i'nin doğrudan kullandığı öznitelik sayısıdır. Bir metot hiçbir öznitelik kullanmıyorsa, ağırlık değeri sınıfta bulunan özniteliklerin sayısına eşit olur (Şekil 4.2).

$$WM(m_i) = \begin{cases} |A(c)|, eger \quad A_{DR}(m_i) = \phi \cup A_{DW}(m_i) = \phi \\ \quad aksi \quad halde \quad |A_D(m_i)| + \\ \quad \sum_{a_j \in A_{DW}(m_i)} \left(\begin{array}{l} |M_{DR}(a_j)| + \\ \left(\sum_{m_k \in M_{IR}(a_j)} \frac{1}{M_{SP}(m_k, a_j)} \right) \end{array} \right) + \\ \quad \sum_{a_j \in A_{IW}(m_i)} \frac{|M_{DR}(a_j)|}{M_{SP}(m_i, a_j)} \end{cases} \tag{4.5}$$

Metot ağırlığı tek bir değerden ziyade çeşitli etkileri içinde barındıran bir değer olarak karşımıza çıkmaktadır. İçerisinde doğrudan erişilen öznitelik sayısını, yazma etkileşimi ile eriştiği özniteliklere erişen metot sayılarını ve dolaylı olarak eriştiği

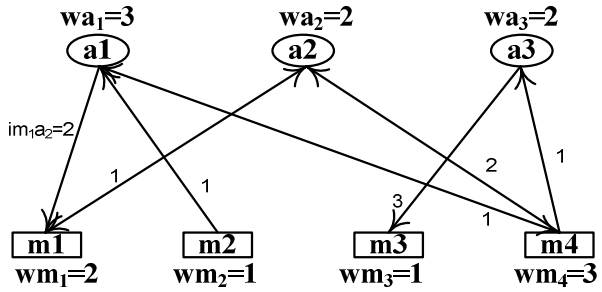
özniteliklerin etkisini içermektedir. Bu değerler arasındaki ilişkiler metot ağırlığı olarak elde edilmektedir. Metot yazma etkileşimi ile bir özniteliğe eriştiğinde, özniteliği okuyan diğer metotlara arasında dolaylı bir etkileşim meydana gelmektedir. Yazılan özniteliği okuyan metotlara arada bir erişim oluştuğundan dolayı yazan metot ağırlığı okuyan metotlar kadar arttırılır. Dolaylı olarak özniteliğe erişilmesi durumunda ise M_{SP} yolu üzerinden özniteliğe erişildiği düşünülmekte ve yol kadar bu etki zayıflatılmaktadır. Erişim yolu ne kadar uzar ise ağırlığa etki o derece azalmaktadır. Bunun yanında eğer, dolaylı erişilen özniteliğe yazma etkileşim tipi ile erişiliyor ise, metot ağırlığına normal şartlarda eklenen metot sayısı M_{SP} yolunun uzunluğu kadar zayıflatılır. Buradan da görüleceği gibi metot ağırlığı kendi içinde birden fazla ölçütü taşıyan bir değerdir ve sınıf içinde uyumu etkileyecek ana unsurları kendi içinde barındırmaktadır. Bu şekilde hesaplanan metot uyumları ise sonuç olarak daha hassas sınıf uyumuna işaret etmektedir.

a_i için öznitelik ağırlığı, $WA(a_i)$, a_i'ye doğrudan erişen metot sayısıdır. Eğer bir öznitelik hiçbir metot tarafından kullanılmıyorsa, ağırlık değeri sınıfta bulunan metotların sayısına eşit olur (Şekil 4.2). Bunun yanında öznitelik ağırlığı, bağımlı değişken etkisini de içinde barındırmaktadır. Bir öznitelik diğer özniteliklerin değerlerini hesaplamada ne kadar fazla yer alıyorsa, bağımlılık etkisi o kadar artmaktadır.

$$WA(a_i) = \begin{cases} |M(c)|, eger \quad M_D(a_i) = \phi \\ aksi \quad halde \quad |M_D(a_i)| + |A_{DA}(a_i)| \end{cases} \tag{4.6}$$

m_i metodu için a_j özniteliğine olan etkileşim sayısı, $I_{MA}(m_i, a_j)$, m_i içerisinden a_j özniteliğine kaç kez erişildiğinin sayısıdır. Bu sayı erişim tipleri olan okuma ve yazma erişimlerinin toplamı olarak tanımlanmaktadır. Eğer m_i ile a_j arasında bir erişim dolayısı ile etkileşim yok ise etkileşim sayısı 1 değerini alır. Ayrıca etkileşim sayısı, mi metodu yazma etkileşimi ile ai özniteliğine eriştiğinde, ai özniteliğini okuyan metot sayılarını da içermektedir. Bu tür bir etki, metot doğrudan özniteliğe erişiyorsa hesaplanmaktadır. Aksi halde arada bir erişim olmadığından etkileşim sayısına bir katkı olmamaktadır. Fakat metot ağırlığı ya da öznitelik ağırlığı ilgili durumdan dolayı etkilenmektedir. Bu etki yukarıda anlatılmıştır.

$$I_{MA}(m_i, a_j) = \begin{cases} 1, eger \quad im_i a_j = 0 \\ aksi \quad halde \quad im_i a_j + \sum_{a_j \in A_{DW}(m_i)} \left| M_{DR}(a_j) \right| \end{cases} \tag{4.7}$$



Şekil 4.4: WM, WA ve I_{MA} örnekleri

Şekil 4.4, bir metot-öznitelik kullanım grafiğindeki üyelerinin ağırlık dağılımlarını ve etkileşim sayılarını göstermektedir. Örneğin, m_1 metodu a_1 özniteliğine 2 kez erişmektedir ve erişim tipi özniteliğe yazma şeklindedir. m_2 metodu da a_1 özniteliğine 1 kez erişmektedir ve erişim tipi, özniteliği okuma şeklindedir.

Ağırlıklar, üyelerin herhangi bir erişimi olmadığı durumda azami değer olarak belirlenmektedir. Bu durum uyuma ters bir etki yaparak onu sınırlandırmaktadır. Çünkü bu durumda olabilecek en yüksek sınıf uyum değeri normal şartlar altındaki uyum değerinden daha az olmalıdır. Yapılan iş bunu temin etmektedir. Ayrıca erişim olmadığı halde etkileşim sayısının bir değerini alması, sınıf uyumunu baskılamak içindir.

Tanım 12. Bir m_i metodu için, metot uyum katkısı, $M_{CC}(m_i)$, sınıfın içinde bulunduğu şartlar altında m_i metodunun genel uyuma olan katkısını göstermektedir.

$$M_{CC}(m_i) = WM(m_i) \times \sum_{a_j \in \left(\begin{smallmatrix} A_D(m_i) \cup \\ A_I(m_i) \end{smallmatrix} \right)} \left(WA(a_j) / M_{SP}(m_i, a_j) \right) + \sum_{a_j \in A_D(m_i)} I_{MA}(m_i, a_j) \tag{4.8}$$

Tanım 13. m_i için temel metot uyumu, $BCOM(m_i)$, m_i'nin uyuma katkı değeri ile m_i'nin içinde bulunduğu sınıfın uyum böleni değerinin oranı şeklinde ifade

edilmektedir. Eğer bir metodun doğrudan eriştiği öznitelik kümesi boş küme ise BCOM değeri sıfıra eşit olur.

$$\mathrm{BCOM}(\mathrm{m_i}) = \begin{cases} 0, eger\ A_D(m_i) = \phi \\ M_{CC}(m_i)/C_{CD}(c), aksi\ \ halde \end{cases} \tag{4.9}$$

Sınıf üyeleri arasındaki erişimler arttıkça etkileşim deseni de yoğunlaşır ve üyelerin uyuma katkı güçleri fazlalaşır. Bunun yanında üyeler arasındaki etkileşim sayılarının artması da üyelerin uyuma katkısını arttırmaktadır. Basit bir sayma işleminden ziyade etkileşim deseni, etkileşim sayıları, etkileşim tipleri ve bağımlı özniteliklerin dikkate alınması uyumun daha etkin bir şekilde elde edilmesini sağlamaktadır. Bir metodun uyumu ayrıca özniteliklerin dolaylı olarak kullanımından da etkilenmektedir. Bir özniteliğin bir metot tarafından dolaylı olarak kullanılıyor olması için, o nesne değişkenine o metodun başka metot çağrımları ile ulaşması gerekmektedir. Bu durum, sınıf karakteristiklerinden bir diğerini göstermektedir. Sınıf içindeki metotlar bazen işlevlerini diğer metotları çağırarak gerçekleştirirler. Örneğin, gelen bir parametre değerine göre sınıf içindeki diğer metotları çağırabilirler. Kendi içlerinde öznitelik kullanmamış olabilirler. Fakat çağrılan metotlar özniteliklere erişiyor olabilir. Metotları çağıran metot bu şekilde dolaylı olarak çağırdığı metotların kullandıkları özniteliklere ortak olurlar. Bu durum doğrudan çağrıma göre daha az uyuma işaret ederler. Fakat durum ne olursa olsun sınıf karakteristiklerine uyum sağlama açısından bu durumda üretilecek ölçüt için sağlanması gereken bir koşuldur.

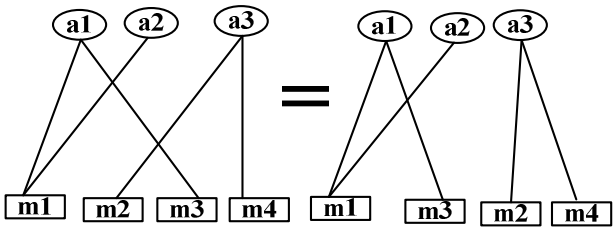
G_{MA}, üyeler arasındaki etkileşimlerin bir gösterimidir. Bazen, bir sınıf ayrık etkileşim desenine sahip olur, böylece G_{MA} içinde farklı ayrık gruplar oluşur. Diğer bir deyişle, sınıfın bazı üyeleri sınıfın diğer üyeleri ile etkileşimde bulunmaz. Bu sebeple, G_{MA} alt metot-öznitelik kullanım grafiklerine parçalanabilir. Doğal olarak, grafiğin içinde fazla grup olduğu durum istenmeyen bir durumdur ve uyumu azaltır. Bir sınıf içerisinde ayrık grupların oluşması tek sorumluluk tasarım prensibine aykırı bir durumdur. Bu açıdan kesinlikle değerlendirilmelidir. Çünkü nesneye yönelik sistemlerin karakteristiklerinden biride tek sorumluluk prensibidir. Bu istenmeyen durum, uygun şekilde değerlendirilmez ise elde edilen uyum değeri yanıltıcı

olacaktır. Oluşan her grup kendi içinde tam uyumlu olsa da uyum değeri doğal olarak azaltılmalıdır. Sağlanan ölçüt bunu kesinlikle sağlamalıdır. Önerdiğimiz ölçüt bunu uygun şekilde yerine getirebilmektedir.

Tanım 14. m_i için ayrık grup etkisi, BCOM*(m_i), ayrık grafik oluştuğunda m_i'nin ayrık grafik üzerindeki BCOM değerine karşılık gelmektedir. Sınıf içinde ayrık gruplar, $D_G(c)$ şeklinde gösterilmektedir. Eğer G_{MA}'da sadece bir grup tanımlı ise, tüm metotların BCOM* değerleri 1 değerine eşit olur.

Her $D_G(c)$ içindeki her metot için, BCOM* değeri hesaplanır. Bir metot için BCOM* değerinin hesaplanması, tek gruplu G_{MA} içindeki metot için yapılan hesaplamadan farksızdır. Aradaki tek fark, hesaplama işlemine katılan metot sayıdır. Sonuç olarak, m_i için BCOM*, (5) kullanılarak bulunabilir. Şekil 4.5, ayrık gruplara sahip bir G_{MA}'yı göstermektedir.

$$\mathrm{BCOM}^*(\mathrm{m_i}) = BCOM(m_i) \quad in \quad D_G(c) \tag{4.10}$$



Şekil 4.5: G_{MA} içindeki farklı gruplar

Tanım 15. m_i için metot uyumu, COM(m_i), metodun son uyum değerini gösterir. m_i'nin BCOM değeri ile BCOM* değerinin çarpımı şeklinde tanımlanır.

$$\mathrm{COM}(\mathrm{m_i}) = \mathrm{BCOM}(\mathrm{m_i}) \times \mathrm{BCOM}^*(\mathrm{m_i}) \tag{4.11}$$

Tanım 16. c_i için sınıf uyumu, ROC(c_i), sınıfın üyeleri arasındaki ilişki derecesidir. Sınıfta bulunan tüm üye metotların uyum değerlerinin toplamı şeklinde tanımlanır.

$$\mathrm{ROC}(c_i) = \begin{cases} 0, eger \\ A_D(m) = \phi \wedge A_I(m) = \phi, \forall m \in M(c) \\ \sum_{m_k \in M(c_i)} COM(m_{k),aksi\ \ durumda} \end{cases} \quad (4.12)$$

Etkileşim sayıları, metot üyelerine göre daha fazla değişime eğilimli olduklarından ve sayı olarak çok fazla değer alabileceklerinden dolayı, uyumun belirli değerler arasında sınırlandırılması için önemli bir sorun teşkil etmektedir. Bu sorunu ortadan kaldırmak için, önerilen yaklaşımın, olası birbirini takip eden iki erişimin eklenmesi durumunda mevcut uyum değerlerinin arasını dolduracak şekilde sürece etki etmesi sağlanmıştır. Uyum ara değerleri bu şekilde doldurulduğundan dolayı, daha hassas metot ve sonuç olarak daha hassas sınıf uyumları elde etmek mümkün hale gelmiştir.

Etkileşim sayılarının artışı ve etkileşim tiplerindeki farklılıklar ile oluşan uyum artışı, sınıf üyeleri arasındaki tüm erişimler mevcut olduğunda sona ermektedir. Bu durumda sınıf uyumu her zaman 1 değerini almaktadır. Tüm erişimler mevcut olduğunda etkileşim sayılarındaki artış sadece metotların genel sınıf uyumundan aldıkları pay üzerinde bir değişiklik yapmaktadır. Böylece tüm erişimleri mevcut metotların aynı uyum değeri alması engellenmiş olmaktadır. Bu noktada metot uyumları incelenerek, metotlardaki anormallik ve bundan dolayı sınıf uyumundaki azalma tespit edilebilmektedir. Burada dikkat edilmesi gereken nokta, metodun özniteliklerle olan etkileşimlerinin sayısı artar ise ve etkileşim olarak yazma etkileşim tipine sahip olursa diğer metotlara göre uyumu artmaktadır. Böylece metotların uyumları birbirlerinden farklı olabilmekte ve daha hassas elde edilebilmektedir. Sonuç olarak, bu sınıf uyumunun daha hassas olarak hesaplanmasına yardımcı olmaktadır.

4.3. Geliştirilen Ölçütün Teorik Olarak Değerlendirilmesi

4.3.1. Teorik değerlendirme 1

Bu kısımda, ROC, Briand tarafından tanımlanmış dört uyum özelliğine göre teorik olarak değerlendirilecektir [61]. Tanımlanmış olan dört özellik uyumu sezgisel ve dikkatli bir şekilde karakterize eden önerilerden biridir. Bu özelliklerin bir uyum

tarafından sağlanıyor olması ölçütün faydalı olduğunu göstermemektedir. Fakat bu özelliklerin sağlanmaması ölçütün kötü tanımlı olduğunu kesin olarak göstermektedir. Aşağıdaki özellikler verilirken "Uyum" aday ölçüyü göstermektedir. "İlişkiler" uyum ölçüsünün odaklandığı bağlantıları göstermektedir. Bir c sınıfı için ilişkiler kümesinin R_C olarak gösterildiğini düşünelim. c sınıfı içindeki tüm muhtemel ilişkiler olduğunda R_C'nin en yüksek olduğu söylenebilir. Bu durumda yeni bir ilişki sınıfa eklenemez. Özellikler ve geliştirilen ölçüte dair özelliklerin nasıl karşılandığı aşağıda yer almaktadır:

- Negatif olmama ve normalleşme: Bir sınıfın uyumu belirli bir aralıkta olmalıdır. [Uyum(c) Є [0, Maksimum]]

Bir sınıf için ROC, 0 ile 1 arasındadır. Sınıf üyeleri arasında hiçbir ilişki oluşmadığında özellikle tanım 2 ve 3'te yer alan kümeler boş küme olacağından uyum değeri 0 olacaktır. R_C en yüksek değere ulaştığında özellikle tanım 10 ve 12'de yer alan sınıf uyum böleni ve sınıf içindeki metotların toplam uyum katkısı birbirlerine eşit olacağı için uyum değeri 1 değerini alacaktır.

- Boş ve maksimum değer: Bir sınıfın uyumu R_C boş olduğunda boş değerdir ve bir sınıfın uyumu R_C en yüksek değerine eriştiğinde uyum maksimum değerini alır. $R_C = Ø \rightarrow$ Uyum(c)=0;R_C = en yüksek → Uyum(c)=maksimum

$$A_D(m) = \phi \wedge A_I(m) = \phi, \forall m \in M(c) \Rightarrow ROC(c) = 0$$
$$|A_D(m)| = \max, \forall m \in M(c) \Rightarrow ROC(c) = 1$$

Etkileşim sayılarındaki değişim, birbirini takip edecek şekilde eklenebilecek iki erişim sonucu elde edilen iki sınıf uyumu değeri arasını dolduracak şekilde sürece katıldığından dolayı, sınıf uyumu negatif olmamakta, normalleşme kuralı bozulmamakta ve maksimum değer belirli bir değerde tutabilmektedir.

- Monotonluk: c Є C olduğunu düşünelim. c sınıfına bazı ilişkiler eklenerek elde edilen sınıf c' olsun. Sonuç olarak $R_C \subseteq R_{C'}$ olacaktır. Bu durumda Uyum(c) ≤ Uyum(c') olmalıdır.

Burada iki durum söz konusudur: Üyeler arasına erişim eklenmesi veya üyeler arasındaki erişimin etkileşim sayı değerinin arttırılması. Bir sınıfın üyeleri arasına yeni bir erişim eklenirse veya olan erişimin etkileşim sayısı artar ise, elde edilen sınıfın uyumu daha fazla olmalıdır. Bir sınıfa bir erişim eklendiğinde, ilgili metot için A_D kümesi artar. Sırası ile tanım 2, 10, 11, 12, 13, 15 ve 16 kullanılarak hesaplama yapılırsa ROC değerinin arttığı gözlenir. Bunun yanında üyeler arasındaki etkileşim sayısı artar ise I_{MA} değeri artar. Bu artış sınıf uyum bölenini arttırdığı gibi metot uyum katkısını da arttırır. Neticede bölendeki artış metot uyumundaki artıştan az kaldığı için metot uyumu dolayısı ile genel sınıf uyumu artar. Fakat iletişim sayısı arttığında metotların uyumları tekrar düzenlenir, iletişim eklenen metot uyumu görece olarak diğerlerinden belirli miktar uyumu kendi üzerine alır. Fakat bu durum genel sınıf uyumunun artmasını engellemez. Sırası ile tanım 5, 10, 11, 12, 13, 15 ve 16 kullanılarak hesaplama yapılırsa ROC değerinin arttığı gözlenir. Erişim tiplerinin değişmesi ve bağımlı öznitelikler metot uyumuna etki etmektedir. Fakat durum ne olursa olsun sınıf uyumuna olan etki bir erişim ya da etkileşim eklendiğinde artma yönünde olmaktadır.

- Bağlantısız sınıfların birleştirilmesi: c_1, c_2 Є C olduğunu düşünelim. c_1 ve c_2 sınıflarının birleşimi ile oluşan sınıf c’ olsun. Bu durumda maksimum{Uyum(c_1), Uyum(c_2)} ≥ Uyum(c’) olmalıdır.

Aralarında hiç bir etkileşim olmayan iki sınıf yeni bir sınıf olacak şekilde birleştirildiğinde, yeni sınıfın uyumu azalmalıdır. Bağlantısız sınıflar bir sınıf içinde birleştirildiklerinde, yeni sınıf içinde farklı gruplar oluşacaktır. Sonrasında, BCOM* değeri, tanım 14 kullanılarak, yeni sınıftaki tüm metotların uyumunu etkileyecektir. Bu etki tüm metotların uyumunu göreceli olarak azaltacak, neticede ROC azalacaktır.

4.3.2. Teorik Değerlendirme 2

Kitchenham, Pfleeger ve Fenton yazılım ölçütlerini değerlendirmek için bir çatı önermişlerdir [62]. Bu çatı kapsamında herhangi bir ölçütün yapısı tanımlanmaktadır. Yapı olarak, analiz edilecek varlıklar (örneğin sınıflar), ölçülecek özellikler (örneğin boyut), kullanılacak birim (örneğin satır sayısı) ve veri ölçeği (örneğin değersel, sırasal, aralık veya oran) tanımlanmalıdır. Birimler sadece aralık veya oran değerleri için geçerlidir. Bir değerin anlamı olabilmesi için varlık, ölçülecek özellik ve birim tanımlı olmalıdır. Ölçüt, izin verilen bir değer kümesi (ayrık ya da sürekli) üzerinde tanımlı olmalıdır.

Bir ölçüt, geçerli olabilmesi için şu özelliklere sahip olmalıdır:

- Özellik doğruluğu: Analiz edilen varlık özelliklere sahip olmalıdır. Çalışmamızda analiz edilecek varlık sınıftır. Sınıflarda kendi içinde uyum, bağımlılık ve boyut gibi özelliklere sahiptir.

- Birim doğruluğu: Birim özellik için uygun olmalıdır. Çalışma kapsamında ölçülecek özellik sınıf uyumudur. Bu açıdan bakıldığında, birim üyeler arasındaki ilişkiler olarak kabul edilecektir. Ayrıca veri ölçeği değerlerden oluşmaktadır.

- Araç doğruluğu: Aracın altında yatan model geçerli olmalı ve araç ayarlanmış olmalıdır. Önerilen uyum ölçütü kendi içinde matematiksel model olarak tutarlıdır. Ölçütün ortaya çıkardığı uyum değerleri belirli değer aralığında olacak şekilde düzenlenmiştir. Bu özellik, tezin ana amaçlarından birini işaret etmektedir.

- Yöntem doğruluğu: Ölçüm için kullanılan yöntem geçerli olmalı ve hatalardan arındırılmış olmalıdır. Önerilen ölçüt mevcut ölçütlerin aksayan kısımlarını iyi belirlenmiş yöntemlerle kapadığı için yöntem olarak geçerlidir. Bu özellik de, tezin ana amaçlarından birini işaret etmektedir.

Bunların dışında, teorik olarak geçerleme için bir ölçüt aşağıdaki özelliklere sahip olmalıdır:

- Ölçüt farklı varlıklar için özellik farklı değerler alabilmelidir.
- Ölçüt seçilen özelliğe göre mantıklı şekilde çalışmalı ve farklı varlıklar için mantıklı değerler üretmelidir.
- Ölçüt farklı varlıklar için özellik benzer değerler alabilmelidir.

Yukarıda belirtilen 3 özellik kapsamında, ROC ölçütü farklı sınıflar için farklı uyum değerleri elde edebilirken, farklı sınıflar için aynı uyum değerlerini de elde edebilmektedir. Önerilen yaklaşım içerisinde metot uyumuna dair fazla ölçüt içerdiğinden metot uyumlarının aynı çıkma olasılığı düşmektedir. Böylece kaba bir uyum hesabı yanında daha hassas bir uyum hesabının yapılabilmesine imkân vermektedir. Fakat bu durum, farklı sınıflar için aynı uyum değerini ya da farklı sınıflar için farklı uyum değerlerini elde etmeye engel değildir.

Bu bölümde geliştirilen ölçüt hakkında tanımlamalar yapılmış ve ölçütün matematiksel modeli çıkarılmıştır. Bu bölüm bu açıdan büyük önem taşımaktadır. Mevcut kabul görmüş ölçütlerden farklı olan ayrıntılar bu kısımda aktarılmıştır. Yapılan bu çalışma sonucu elde edilen ölçüt tanımı bu noktadan sonra diğer bölümlerde deneysel olarak sınanacaktır.

5. DENEYSEL ÇALIŞMALAR

5.1. Giriş

Bu bölümde, yapılmış olan deneysel çalışmalar hakkında bilgi aktarılmaktadır. İstenilen amaca ulaşıldığını gösterebilmek için üç ayrı deneysel çalışma yapılmıştır. Elde edilen sonuçlar farklı kapsamlarda ayrı olarak değerlendirilmiştir. Yapılan her bir deneysel çalışma özellikle farklı noktalara değinmektedir.

5.2. Deneysel Çalışma 1

Şekil 5.1 C# ile kodlanmış Stack sınıfını göstermektedir. Stack sınıfı altı metot içermektedir. Sınıf içerisindeki özel metotlar; *Stack* yapıcı metodu ve *Size* erişim metodudur.

```
Class Stack {
  private int top;
  private int size;
  private int[] array;
  Public Stack(int size){
    this.size = size;
    array = new int[size];
    top = 0;
  }
  public bool IsEmpty(){ return (size==0); }
  public int Size()      { return size;  }
  public int Peek(){ return array[top]; }
  public void Push(int item){
    if(top!=size) array[top++]=item;
    else Console.WriteLine("Stack is Full. \n");
  }
  public int Pop(){
    if(!IsEmpty()) --top;
  }
}
```

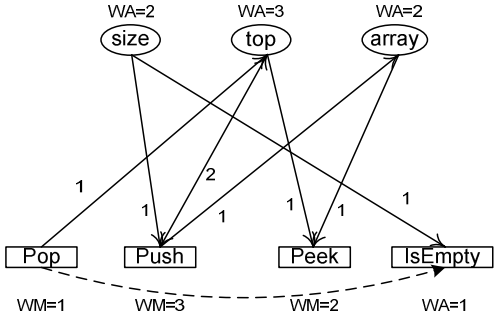
Şekil 5.1: Stack sınıf örneği

Uyum hesaplama süreci işletilmeden önce, sınıf içinde özel metotlarla ilgili bazı ön işlemler yapılmaktadır. Örneğin erişim ve delegasyon metotları kendi eriştikleri öznitelikleri kendilerini çağıran metotlara geçirirler. Böylece, bu öznitelikler ilgili metotlar tarafından doğrudan erişilmiş olarak kabul edilirler. Bu işlemin amacı özel metotların uyumu kötü yönde etkilemesinin önüne geçmektir. Şekil 5.2 üyeler arasındaki sonuç etkileşimlerini G_{MA} grafiği olarak göstermektedir.

Tanım 4.2'ye göre,
A_D(Peek)={top, array},
A_D(Pop)={top},
A_D(Push)={size, top, array},
A_D(IsEmpty)={size},
A_I(Pop)={size} elde edilir.
Tanım 4.3'e göre,
M_D(top)={Pop, Push, Peek},
M_D(array)={Push, Peek},
M_D(size)={Push, IsEmpty} elde edilir.
Tanım 4.5'e göre,
I_{MA}(IsEmpty, size)=1,
I_{MA}(Peek, array)=1,
I_{MA}(Peek, top)=1,
I_{MA}(Push, top)=2,
I_{MA}(Push, size)=1,
I_{MA}(Push, array)=1,
I_{MA}(Pop, top)=1 olarak elde edilir.
Tanım 4.6'ya göre,
M_{DC}(Pop)={IsEmpty} olarak elde edilir.
Tanım 4.9'a göre,
M_{IP}(Pop,size)={(Pop,IsEmpty)} ve M_{SP}(Pop,size)=2 olarak elde edilir.
Bu bilgilerden hareket ile Tanım 4.10'e gore,
C_{CD}(Stack)=97 olarak elde edilir.
Metot ve öznitelik ağırlıkları aşağıdaki şekilde elde edilirler:
WA(top)=3, WA(size)=2, WA(array)=2,

WM(IsEmpty)=1, M_{CC}(IsEmpty)=3,
BCOM(IsEmpty)=0.0309,
WM(Peek)=2,M_{CC}(Peek)=12, BCOM(Peek)=0.1237,
WM(Push)=6,M_{CC}(Push)=49, BCOM(Push)=0.5051,
WM(Pop)=3, M_{CC}(Pop)=15, BCOM(Pop)=0.1546.

Sonuç olarak, Tanım 4.13'e göre, ayrık gruplar olmadığından dolayı BCOM değerleri COM değerlerine eşit olmaktadır. Tanım 4.16'ya göre, sınıf uyumu ROC(Stack)=0,7525.



Şekil 5.2: Stack G_{MA}

Tablo 5.1 deneysel çalışmanın diğer mevcut ölçütler için ölçüm değerlerini içermektedir. Burada dikkat edilmesi gereken bu ölçütlerin üyeler arasındaki etkileşim sayılarına dikkat etmemeleridir.

Önerilen yaklaşım erişim, etkileşim deseni ve etkileşim sayılarının tümünü uyum sürecine katan ve ayrık grup etkisini dikkate alan bir yaklaşımdır. Oysaki mevcut ölçütler, yukarıda bahsedilen kriterlere tam olarak uymamakta ve bundan dolayı uygun olmayan uyum değerleri üretmektedirler.

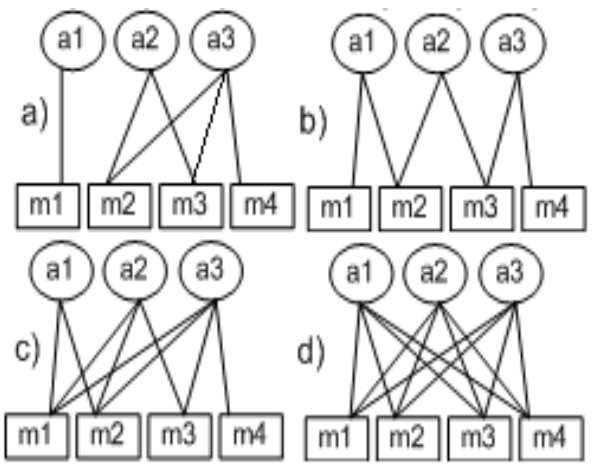
Tablo 5.1: Çalışma durumu ölçüt sonuçları

LCOM1	4	**LCOM4**	1	**Coh**	0,61
LCOM2	0	**Co**	0,5	**LCC**	1
LCOM3	1	**LCOM5**	0,47	**TCC**	0,6

Tablo 5.1'de görülebileceği gibi, LCOM1-2-3-4 ölçütleri normalize edilmemişlerdir. LCOM2 ise sınıf üyeleri bir şekilde etkileşim içinde olduğundan tam uyum değerini

vermiştir. LCOM3-4 ise sınıf üyeleri ilişkide olduğundan tek bir grafik oluştuğunu hesap edip uyum değeri olarak bir değerini vermiştir. LCOM5 metot etkileşimlerini hesap etmediği için daha düşük bir değer vermiştir. Co, grafik tabanlı bir diğer ölçüt olup üyeler bir şekilde etkileşim içinde olduğundan 0-1 aralığında ortalama bir değer hesaplamıştır. Coh sadece metot-öznitelik kullanımına dikkat ettiğinden ve örneğimizde metot çağrımları az olduğundan daha uygun bir sonuç oluşturmuştur. TCC ve LCC göreli olarak daha iyi olmakla birlikte, yukarıda hiçbir ölçüt etkileşim deseni, metot-öznitelik kullanım sayıları, metot-metot çağrımları, etkileşimin okuma ya da yazma olmasına ya da ayrık gruplara tümden dikkat etmemiştir. Tabii ki bu çalışma geliştirilen ölçütün faydalılığını gösterse de tam olarak değerlendirmeye yeterli değildir. Bunun için aşağıda daha kapsamlı bir çalışma gerçekleştirilmiştir.

Tablo 5.2: Uyum ölçüt değerlendirme örneği



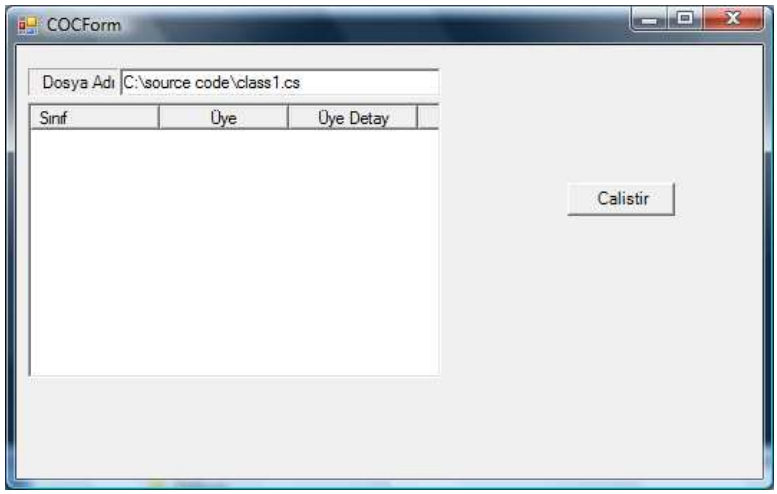
LCOM1	a)3 b)3 c)0 d)0
LCOM2	a)0 b)0 c)0 d)0
LCOM3	a)2 b)1 c)1 d)1
LCOM4	a)2 b)1 c)1 d)1
LCOM5	a)2/3 b)2/3 c)1/3 d)0
Co	a)* b)0 c)1 d)1
Coh	a)1/2 b)1/2 c)1/3 d)1
LCC	a)1/2 b)1 c)1 d)1
TCC	a)1/2 b)1/2 c)1 d)1
ROC	a)0,203 b)0,62 c)0,9 d)1

Tablo 5.2'de 4 sınıf için G_{MA}'lar bulunmaktadır. Sadece a ayrık iki gruba sahiptir. Normal olarak, b, c ve d'nin üyeleri arasındaki iletişimler nedeni ile a'dan daha uyumlu olmaları gerekmektedir. Bununla birlikte bazı ölçütler a ve b arasındaki uyumu ayırt edememektedir. Bazı ölçütler ise üyeleri bir şekilde iletişim içinde olan sınıflara maksimum uyum değerini atamaktadır. Örneğin c, b'ye göre daha sıkı bir ilişki ağına sahiptir. Bu sebeple, c, b'den aynı şekilde d'de c'den daha uyumlu olmalıdır. Bizim yaklaşımımız, hem etkileşim gücüne hem de ayrık grup etkisine uygun uyum değeri belirleyebilmektedir.

5.3. Uyum Hesaplama Yazılımı

Tez kapsamında, mevcut uyum ölçütleri ile ROC için uyum değerlerinin hesaplanmasında kullanılacak, bir uyum ölçüm yazılımı geliştirilmiştir. Uyum hesaplama yazılımı C# uygulamaları için geliştirilmiştir. Bu yazılım, Microsoft Windows XP Servis Paketi 2 platformunda, Microsoft Visual Studio 2005 geliştirme ortamı ile C# programlama dili yardımı ile geliştirilmiştir. Oldukça basit bir arayüze sahiptir. Temel ihtiyaç olunan tüm bilgileri listeleyebilmektedir.

Şekil 5.3'de uyum ölçüm aracının bir görüntüsü yer almaktadır. Uyumun hesaplanabilmesi için kaynak kodun yer aldığı "cs" uzantılı dosyanın adı "Dosya Adı" kutucuğa yazılmalıdır. Takiben "Calistir" düğmesine basıldığında hesaplama süreci başlatılmış olacaktır. Hesaplama esnasında düz yazı dosyasından okunan sınıf üzerinde bazı katar işleme fonksiyonları ile ayrıştırma yapılmaktadır. Elde edilen sonuçlar konsol üzerine ve ekranda yer alan liste kutusuna aktarılmaktadır. Uyum ölçüm yazılımı, öznitelik listesini, metot listesini, özniteliği kullanan metotların listesini, metotların birbirlerine çağrımlarının listesini, erişim sayı listesini ve erişim tipi listesini içermektedir. Bu bilgiler yanında elde edilen metot ağırlıkları, öznitelik ağırlıkları, yardımcı metot uyumu ve nihai olarak sınıf uyumu yer almaktadır.



Şekil 5.3: Uyum ölçüm aracı görünümü

Ölçüm yazılımının oluşturulması sırasında, uyum hesaplama algoritmaları dile bağlı yapılardan ayrı tutulduğundan dolayı ileride gerek görüldüğünde farklı bir nesneye

yönelik programlama dilinde de gerçeklenebilir durumdadır. Sunulan ölçüt yanında, literatürde kabul görmüş olan bazı ölçütler proje kapsamında gerçeklenmiştir. Bu ölçütler şunlardır: LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, Co, Coh ve TCC-LCC.

Yazılan program basit bir arayüz içermektedir. Model bileşeni içerisinde sınıfın alınması, temizlenmesi ve ayrıştırılması gibi işlevler yerine getirilmektedir. Diğer temel bileşenler Sınıf, Metot, Öznitelik ve Etkileşim sınıflarıdır. Bunlar ayrıştırma sonucu oluşan ve ölçüt için gerekli parametreleri sağlayan birimlerdir. Asıl hesaplama işlemlerini gerçekleyen birim ROCDomain sınıfıdır. Sunulan ölçüt ve diğer ölçütler bu sınıf kapsamında elde edilen ayrıştırma verileri yardımı ile hesaplanmaktadır. Şekil 5.4'de yer alan UML diyagramı anlatılan bu genel görünüşü yansıtmaktadır. Tablo 5.3'te yer alan kod alıntısı ise ROCDomain kapsamında sınıfların ROC uyum değer hesabını yapan sözde kodu göstermektedir:

Tablo 5.3: Uyum hesaplama sözde kodu

```
public double ROCHesaplayici()
{
        double WMler=0;
        double WAlar=0;
        double IMAlar=0;

        ArrayList liste;
        ArrayList IMADegerleri;

        //WM'leri hesapla
        liste=sinif.MetotListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                WMler += WMHesapla((Metot)liste[k]);
        }

        //WA'ları hesapla
        liste=sinif.OznitelikListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                WAlar += WAHesapla((Oznitelik)liste[k]);
        }
```

Tablo 5.3 (Devam): Uyum hesaplama sözde kodu

```
//IMA'ları hesapla
        IMAlar = IMAHesapla();

        //CCD' yi hesapla
        double CCD=CCDHesapla(WMler,WAlar,IMAlar);
        sinif.CCDDegerAyarla(CCD);

        //MCC'leri hesapla
        liste=sinif.MetotListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                MCCHesapla((Metot)liste[k]);
        }

        //BCOM'ları hesapla
        liste=sinif.MetotListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                BCOMHesapla((Metot)liste[k]);
        }

        //BCOM*'ları hesapla
        liste=sinif.MetotListesiGetir();

        Metot metodHesap;
        Oznitelik oznitelikHesap;
        Etkilesim etkilesimHesap;

        //yukarıdaki yer alan süreç her bir ayrık grup için tekrarlanmalı
        for (int l=1;l<=sinif.MaxGrupNoDegerGetir();l++)
        {

                //eğer ayrık grup yok ise bu bölümü işleme
                int grupElemanSayisi=sinif.GrupElemanSayisi(l);
                if (grupElemanSayisi==0)
                {
                        continue;
                }

                WMler=0;
                WAlar=0;
                IMAlar=0;
                CCD=0;

                liste=sinif.MetotListesiGetir();
                for (int p=0;p<liste.Count;p++)
                {
```

Tablo 5.3 (Devam): Uyum hesaplama sözde kodu

```
            metodHesap=(Metot)liste[p];

            //eğer metot ilgili ayrık grupta ise
            if (metodHesap.GrupNoDegerGetir()==l)
            {
                  WMler+=metodHesap.WMDegerGetir();
            }
      }


      liste=sinif.OznitelikListesiGetir();
      for (int p=0;p<liste.Count;p++)
      {
            oznitelikHesap=(Oznitelik)liste[p];

            //eğer öznitelik ilgili ayrık grupta ise
            if (oznitelikHesap.GrupNoDegerGetir()==l)
            {
                  WAlar+=oznitelikHesap.WADegerGetir();
            }
      }

      liste=sinif.EtkilesimListesiGetir();
      for (int p=0;p<liste.Count;p++)
      {
            etkilesimHesap=(Etkilesim)liste[p];

            //eğer etkileşim ilgili ayrık grupta ise
            if (etkilesimHesap.GrupNoDegerGetir()==l)
            {
                  IMAlar+=etkilesimHesap.IMADegerGetir();
            }
      }

      CCD=CCDHesapla(WMler,WAlar,IMAlar);

      liste=sinif.MetotListesiGetir();
      for (int p=0;p<liste.Count;p++)
      {
            metodHesap=(Metot)liste[p];

            //eğer metot ilgili ayrık grupta ise
            if (metodHesap.GrupNoDegerGetir()==l)
            {
                  BCOMAyrikGrupHesapla((Metot)liste[p],CCD);
            }
      }
```

Tablo 5.3 (Devam): Uyum hesaplama sözde kodu

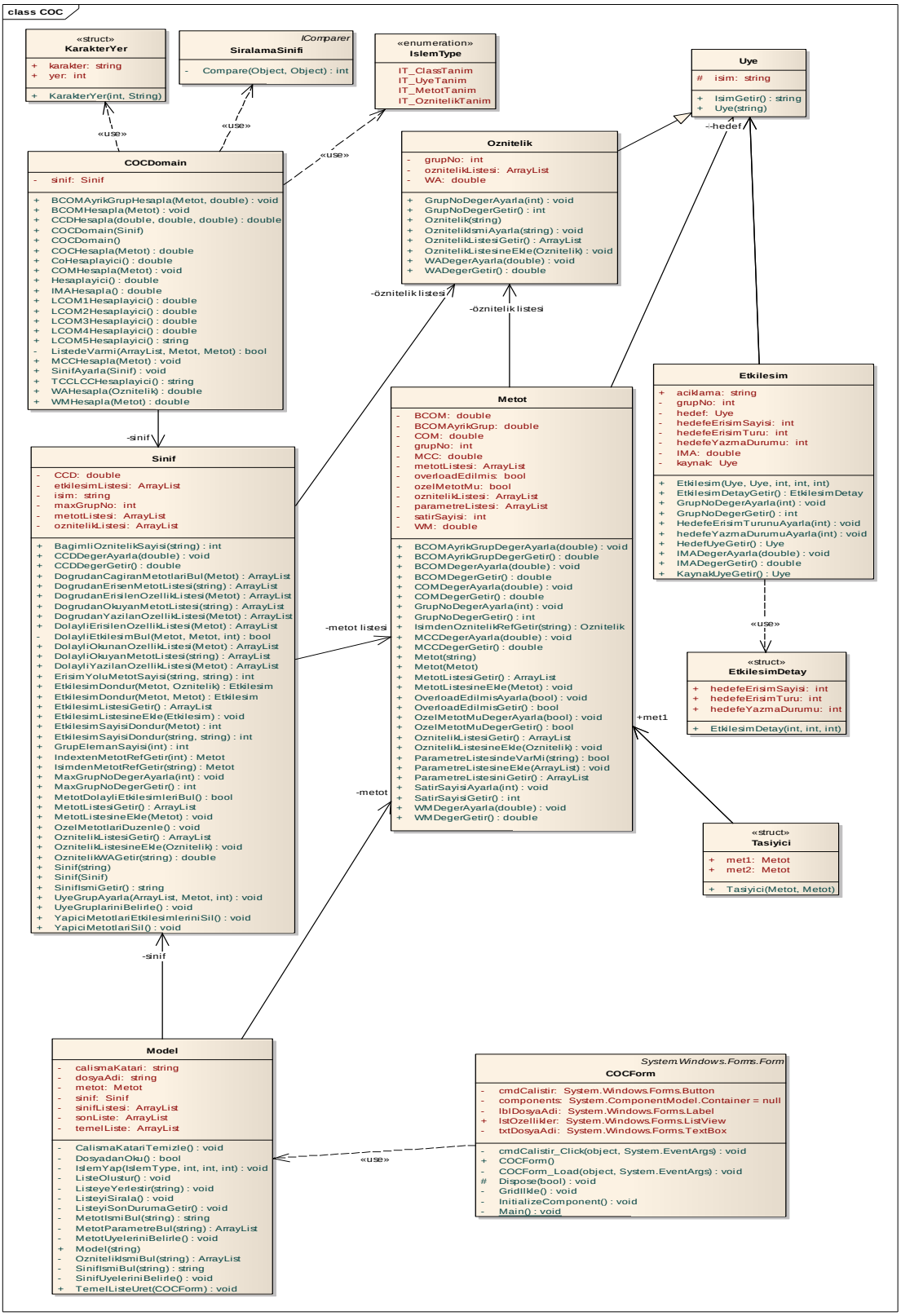
```
}

        //BCOM ları hesapla
        liste=sinif.MetotListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                COMHesapla((Metot)liste[k]);
        }

        //ROC hesapla
        double ROC=0;

        liste=sinif.MetotListesiGetir();
        for (int k=0;k<liste.Count;k++)
        {
                ROC+=ROCHesapla((Metot)liste[k]);
        }

        return ROC;
}
```



Şekil 5.4: UML diyagramı

Aşağıda, uyum ölçüm yazılımı ile elde edilen bir çıktı gösterilmektedir. Çıktı Log4Net sisteminden bir sınıfa dair bilgileri göstermektedir. Bu sistem üzerindeki diğer çalışmalar ilerleyen kısımlarda aktarılmaktadır.

Aşağıdaki çıktı da ROC uyum değerinin gösterildiği satırın altında her bir metot için (özel metotlar dışındaki) metot uyumu değerleri yer almaktadır. Bu değerlerin elde edilmesinde kullanılan öznitelikler ve onlara ait hesaplanmış veriler ise metotların altında listelenmişlerdir. Aşağıda kullanılmış olan kısaltmaların anlamları aşağıda verilmiştir.

WM: Metot Ağırlığı
BCOM: Temel Metot Uyumu
COM: Nihai Metot Uyumu
WA: Öznitelik Ağırlığı
ES: Erişim Sayısı
ET: Erişim Türü (0: Doğrudan, 1: Dolaylı)
YD: Yazma Durumu (1: Yazma, 2: Okuma 3: Okuma/Yazma)

LCOM1: 26
LCOM2: 7
LCOM3: 4
LCOM5: 0,827160493827161
Coh: 0,255555555555556
LCOM4: 3
Co: 0,333333333333333
Özel metotlar haricindeki metotların çağrım listesi başlangıç
ActivateOptions - SendBuffer
ActivateOptions - InitializeDatabaseCommand
OnClose - SendBuffer
OnClose - SendBuffer
OnClose - InitializeDatabaseConnection
OnClose - InitializeDatabaseCommand
SendBuffer - SendBuffer
SendBuffer - InitializeDatabaseConnection
SendBuffer - InitializeDatabaseCommand
AddParameter - SendBuffer

AddParameter - InitializeDatabaseCommand

SendBuffer - InitializeDatabaseConnection

SendBuffer - InitializeDatabaseCommand

InitializeDatabaseConnection - InitializeDatabaseCommand

ActivateOptions - InitializeDatabaseConnection

SendBuffer - GetLogStatement

Özel metotlar haricindeki metotların çağrım listesi son

TCC: 0,388888888888889

LCC: 0,444444444444444

ROC: 0,0443004448109466

ActivateOptions WM:10 BCOM:0,099815157116451 COM:0,0127363385734602

m_usePreparedCommand WA:2

ES:2 ET:0 YD:3

m_commandText WA:3

ES:2 ET:0 YD:1

m_dbConnection WA:6

ES:1 ET:1 YD:3

m_connectionString WA:2

ES:1 ET:1 YD:1

m_dbCommand WA:3

ES:1 ET:1 YD:3

m_commandType WA:2

ES:1 ET:1 YD:1

m_parameters WA:3

ES:1 ET:1 YD:1

OnClose WM:9,5 BCOM:0,0720887245841035 COM:0,00664333709541597

m_dbConnection WA:6

ES:3 ET:0 YD:3

m_dbCommand WA:3

ES:3 ET:0 YD:3

SendBuffer WM:3,5 BCOM:0,0299445471349353 COM:0,00114627047161142

m_dbConnection WA:6

ES:3 ET:0 YD:1

m_useTransactions WA:1

ES:1 ET:0 YD:1

m_dbCommand WA:3

ES:1 ET:1 YD:1

m_usePreparedCommand WA:2

ES:1 ET:1 YD:1

m_parameters WA:3

ES:1 ET:1 YD:1

AddParameter WM:4 BCOM:0,011090573012939 COM:0,000157238747820496

m_parameters WA:3

ES:1 ET:0 YD:2

SendBuffer WM:4 BCOM:0,0325323475046211 COM:0,00135295207013547

m_dbConnection WA:6

ES:1 ET:0 YD:1

m_dbCommand WA:3

ES:4 ET:0 YD:1

m_usePreparedCommand WA:2

ES:1 ET:0 YD:1

m_parameters WA:3

ES:1 ET:0 YD:1

GetLogStatement WM:0 BCOM:0 COM:0

InitializeDatabaseConnection WM:6,5 BCOM:0,0451016635859519 COM:0,00260037946951141

m_dbConnection WA:6

ES:3 ET:0 YD:3

m_connectionString WA:2

ES:2 ET:0 YD:1

ResolveConnectionType WM:1 BCOM:0,0022181146025878 COM:0,0022181146025878

m_connectionType WA:1

ES:2 ET:0 YD:1

InitializeDatabaseCommand WM:8 BCOM:0,116820702402957 COM:0,0174458137804039

m_dbConnection WA:6
ES:1 ET:0 YD:1
m_dbCommand WA:3
ES:14 ET:0 YD:3
m_commandText WA:3
ES:3 ET:0 YD:1
m_commandType WA:2
ES:1 ET:0 YD:1
m_parameters WA:3
ES:1 ET:0 YD:1

5.4. Deneysel Çalışma 2

Sınıf uyumunu elde etmede geliştirilen yeni ölçütün etkinliğini gösterebilmek için, 18 tane farklı özelliklere sahip sınıf oluşturulmuştur. Bu sınıfların her biri ayrı noktalara odaklanmaktadır. Bu sınıflar Tablo 5.3'de verilmiştir. Bir uygulama kapsamında kullanılan sınıflar, ayrıntılarından arındırıldığı zaman temelde üyelere (metotlar ve öznitelikler) ve bu üyeler arasındaki ilişkilere sahiptir. Göz önünde bulundurulan uygulamanın alanı, kapsamı ve tarzı ne olursa olsun bu temel gerçek değişmemektedir. 18 farklı sınıf bu gerçek üzerine ölçütün farklı değişimlere karşı davranışını gözlemleyebilmek için oluşturulmuştur.

Önerilen ölçütlerin değerlendirilmesi için literatürde yapılan çalışmalar özellikle açık kaynak kodlu yazılımların değerlendirilmesi üzerine yapılandırılmıştır. Bu tür bir ölçüt değerlendirme yaklaşımı gerekli olmasının yanında bazı kısıtlara da sahiptir. Çünkü seçilen yazılım ve yazılımın değerlendirilmeden önce üzerinde yapılan temizleme işlemleri sonucu büyük ölçüde değiştirebilmektedir. Özellikle seçilen yazılım, yazılımı yapan grubun temel özelliklerine ve nesneye yönelik düşünceyi ne kadar iyi bildiğine fazlaca dayanmaktadır. Bu açıdan bakıldığında, basit ölçütler nesneye yönelik düşüncenin tam olarak kullanılmadığı yazılımlarda başarılı gibi görünebilir. Bu sınıflar yukarıda bahsedilen soruna objektif yaklaşmayı sağlamak amacı ile oluşturulmuşlardır. Her şey açık olarak belirtildiği için şüpheye mahal vermeden değerlendirme imkânı sunmaktadır. Literatürde böyle bir tekilleştirilmiş

örnek kümesi oluşturma çalışması yapılmamıştır. Oluşturulan sınıflar yapıları itibari ile basit olsalar da böyle bir çalışma bu konuda yapılan ilk çalışmadır. Yapılan bu çalışma ileride örnek kümesi oluşturma çabası için kullanılacaktır. Tablo 5.4 oluşturulan sınıflara ait ayrıntıları içermektedir.

Tablo 5.4: Sınıf tanımları

	Sınıf Tanımları
D1	Üyesi olmayan sınıf
D2	4 metot, 6 öznitelik, etkileşim yok
D3	D2+1 etkileşim
D4	D2+öznitelik kullanımı ile tam etkileşimli
D5	D4+bir metot bir özniteliği kullanmıyor
D6	D4+1 etkileşim eklenmiş
D7	D4+1 etkileşim çıkarılmış
D8	D4+mevcut etkileşime ilave etkileşim
D9	D2+tüm metotlar tüm öznitelikleri kullanır
D10	D9+bir metot bir özniteliğe dolaylı erişiyor
D11	D2+1 ayrık grup
D12	D2+1 metot tüm özniteliklere erişir, 3 metot bu metodu çağırır
D13	D2+1. metot tüm özniteliklere erişir, 2. 1.'ye erişir, 3. 2.'ye erişir, 4. 3.'ye erişir
D14	D2+2 ayrık grup ve kullanılmayan bir öznitelik
D15	D4+1 etkileşim yazma durumunda iken okuma durumuna çevrilir
D16	D4+bağımlı bir öznitelik eklenir
D17	D2+metotlar ve öznitelikler arasında minimum etkileşim(1)
D18	D2+metotlar ve öznitelikler arasında minimum etkileşim(2)

5.4.1. Sınıf tanımları ve detayları

D1: Üyesi olmayan sınıf

Kod :

Tablo 5.5: Durum 1 sınıf kodu

```
public class SampleClass
{
}
```

Uyum Sonuçları:

LCOM1:0

LCOM2:0

LCOM3:0

LCOM5: NaN

Coh: 0

LCOM4:0

Co:1

TCC: NaN LCC: NaN

ROC:0

Yorum: Sonuçlardan da anlaşılacağı gibi boş sınıf için bir uyum sıfır(0) değerini almaktadır. Fakat bazı uyum ölçütlerinde tanımsız olarak karşımıza çıkmaktadır.

D2: 4 metot, 6 öznitelik, etkileşim yok

Kod:

Tablo 5.6: Durum 2 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
```

Tablo 5.6 (Devam): Durum 2 sınıf kodu

```
{                }
        public void method2()
        {                }
        public void method3()
        {                }
        public void method4()
        {                }
}
```

Uyum Sonuçları:

LCOM1:6

LCOM2:6

LCOM3:4

LCOM5: 1,33333333333333

Coh: 0

LCOM4:4

Co:1

TCC: 0 LCC: 0

ROC:0

Yorum: Bir sınıfın uyumundan bahsedebilmek için metot, öznitelik ve bunlar arasında bir etkileşimin olması gerekmektedir. Bu nedenle uyum hala sıfır(0) değerine sahip olmaktadır. LCOM değerleri uyumun olmamasına odaklandığından sıfırdan farklı değer almaları kaçınılmazdır. Burada dikkat edilmesi gereken nokta bazı ölçütlerin bir aralığa normalize edilmediğidir.

D3: D2+1 etkileşim

Kod:

Tablo 5.7: Durum 3 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=5;
                Console.WriteLine("dump1");
                Console.WriteLine("dump2");
                Console.WriteLine("dump3");
                Console.WriteLine("dump4");
        }
        public void method2()
        {               }
        public void method3()
        {               }
        public void method4()
        {               }
}
```

Uyum Sonuçları:

LCOM1:6

LCOM2:6

LCOM3:4

LCOM5: 1,27777777777778

Coh: 0,0416666666666667

LCOM4:4

Co:1

TCC: 0 LCC: 0

ROC:0,00478468899521531

Yorum: Sınıf üyeleri arasında bir etkileşimin bulunması uyumun belirli oranda artmasını sağlamalıdır. Çünkü bu durum üyeler arasında bir amacı gerçekleştirmede bir işbirliğinin olduğunu göstermektedir. Tek bir etkileşimin olması, uyumun çok az bir miktarda artması anlamına gelmektedir. LCOM metotlar arasındaki ortak öznitelik kullanımını temel aldığından dolayı bir önceki duruma göre değişmemektedir. Sadece LCOM5 metot öznitelik kullanımına bağlı olduğundan azalma ile sonuçlanmıştır. Coh değeri LCOM5'in ters yönlü bir sürümü olduğundan artış mantıklıdır. Fakat büyüme olarak bir etkileşim oldukça fazla bir artışa sebep olmaktadır. ROC değeri de artış göstermektedir, fakat büyüme tek bir etkileşim olduğundan mantıklı seviyededir.

D4: D2+öznitelik kullanımı ile tam etkileşimli

Kod:

Tablo 5.8: Durum 4 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute4=4;
        }
```

Tablo 5.8 (Devam): Durum 4 sınıf kodu

```
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {
                attribute1=1;
                attribute3=3;
                attribute4=4;
                attribute6=6;
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
        }
}
```

Uyum Sonuçları:

LCOM1:1

LCOM2:0

LCOM3:1

LCOM5: 0,722222222222222

Coh: 0,458333333333333

LCOM4:1

Co:0,666666666666667

TCC: 0,66 LCC: 1

ROC:0,606060606060606

Yorum: Sınıf özellik ortak kullanımı ile tam bağlı bir hale getirildiğinde uyum değerleri hissedilir bir şekilde artmaktadır. Ortalama üzerine çıkan uyum değerleri

bir şekilde üyelerin birbirlerine bağlı olduğunu göstermektedir. Fakat bazı ölçütler bu artışı yüksek bir değer ile ifade etmektedir. Bu durum istenmeyen bir durumdur.

D5: D4+bir metot bir özniteliği kullanmıyor

Kod:

Tablo 5.9: Durum 5 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute2=2;
                attribute4=4;
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {
                attribute3=3;
                attribute4=4;
                attribute6=6;
        }
        public void method4()
```

Tablo 5.9 (Devam): Durum 5 sınıf kodu

```
        {
            attribute4=4;
            attribute5=5;
        }
    }
```

Uyum Sonuçları:

LCOM1:1

LCOM2:0

LCOM3:1

LCOM5: 0,833333333333333

Coh: 0,375

LCOM4:1

Co:0,666666666666667

TCC: 0,5 LCC: 0,83

ROC:0,420634920634921

Yorum: Ortak kullanılan bir özniteliğe olan etkileşimin kaldırılması, aynı zamanda özniteliği doğrudan kullanan metotların etkileşimini de azalmaktadır. Doğal olarak bu durum uyumu azaltmalıdır. Çünkü metot ve öznitelik arasında olan bağlar bir şekilde koparılmış daha az bağlı bir durum ortaya çıkarılmıştır. ROC değerinde bu durum görülebilmektedir. Fakat doğrudan metot iletişimine dayanan uyum ölçütlerinde bir azalma olmamaktadır. Çünkü çıkarılan öznitelik zaten hala ortak öznitelik paylaşan metotlar arasından çıkarılmıştır. Böylece, çıkarılan etkileşimlerden etkilenmemişlerdir. Bu durum istenmeyen bir durumdur.

D6: D4+1 etkileşim eklenmiş

Kod:

Tablo 5.10: Durum 6 sınıf kodu

```
public class SampleClass
{       private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute4=4;
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {       attribute1=1;
                attribute3=3;
                attribute4=4;
                attribute6=6;
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
}
```

Uyum Sonuçları:

LCOM1:0

LCOM2:0

LCOM3:1

LCOM5: 0,666666666666667

Coh: 0,5

LCOM4:1

Co:1

TCC: 0,83 LCC: 1

ROC:0,641025641025641

Yorum: Mevcut olan sınıfın üyeleri arasında ek bir etkileşim oluşturulduğunda doğal olarak uyumun artması gerekmektedir. Çünkü üyeler arası bağlar bir şekilde güçlendirilmiş olmaktadır. LCOM2 gibi ölçütlerde bu durum görülememektedir. Ayrıca bazı ölçütlerde, metotlar arasında herhangi bir bağlantı var iken ek etkileşim eklendiğinde bu durum tam olarak hesaplanamamaktadır. Bu istenmeyen bir durumdur. Fakat genel olarak uyum artma yönündedir.

D7: D4+1 etkileşim çıkarılmış

Kod:

Tablo 5.11: Durum 7 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
```

Tablo 5.11 (Devam): Durum 7 sınıf kodu

```
                attribute1=1;
                attribute2=2;
                attribute4=4;
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {
                attribute1=1;
                attribute3=3;

                attribute6=6;
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
        }
}
```

Uyum Sonuçları:

LCOM1:2

LCOM2:0

LCOM3:1

LCOM5: 0,777777777777778

Coh: 0,416666666666667

LCOM4:1

Co:0,333333333333333

TCC: 0,66 LCC: 1

ROC:0,518181818181818

Yorum: Sınıfın üyeleri arasındaki bir etkileşimin çıkarılması doğal olarak uyumun azalmasına sebep olmaktadır. Çünkü üyeler arasındaki bağlar bir şekilde zayıflamış olmaktadır. LCOM2 gibi ölçütlerde bu durum görülememektedir. Fakat genel olarak uyum azalma yönündedir.

D8: D4+mevcut etkileşime ilave etkileşim

Kod:

Tablo 5.12: Durum 8 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute4=4;
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {
```

Tablo 5.12 (Devam): Durum 8 sınıf kodu

```
                attribute1=1;
                attribute3=3;
                attribute4=4;
                attribute6=6;
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
                bool temp=false;
                if (temp==false)
                {
                        attribute5=7;
                }
        }
}
```

Uyum Sonuçları:

LCOM1:1

LCOM2:0

LCOM3:1

LCOM5: 0,722222222222222

Coh: 0,458333333333333

LCOM4:1

Co:0,666666666666667

TCC: 0,66 LCC: 1

ROC:0,609022556390977

Yorum: Metotlar öznitelikleri kullanmaktadırlar. Bir metot bir özniteliği bir veya daha fazla kullanabilmektedir. Birden fazla kullanım metot ile öznitelik arasında yoğun bir etkileşim olduğunu göstermektedir. Doğal olarak bu durum uyumun belirli ölçüde artmasına sebep olmalıdır. Çünkü üyeler arası mevcut olan bağlılık artış

göstermektedir. ROC haricindeki diğer ölçütler bu duruma yeteri kadar özen göstermemektedir. Metot-Öznitelik arası etkileşim yoğunluğuna dikkat edilmemektedir.

D9: D2+tüm metotlar tüm öznitelikleri kullanır

Kod:

Tablo 5.13: Durum 9 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method2()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
```

Tablo 5.13 (Devam): Durum 9 sınıf kodu

```
                attribute5=5;
                attribute6=6;
        }
        public void method3()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method4()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
}
```

Uyum Sonuçları:

LCOM1:0

LCOM2:0

LCOM3:1

LCOM5: 0

Coh: 1

LCOM4:1

Co:1

TCC: 1 LCC: 1

ROC:1

Yorum: Tüm metotlar ile tüm öznitelikler arasında tam etkileşim olduğunda uyum en yüksek değerini almaktadır. Sonuçlardan da görüleceği gibi tüm ölçütler için bu duruma özen göstermektedir.

D10: D9+bir metot bir özniteliğe dolaylı erişiyor

Kod:

Tablo 5.14: Durum 10 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method2()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
```

Tablo 5.14 (Devam): Durum 10 sınıf kodu

```
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method3()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method4()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                method3();
        }
}
```

Uyum Sonuçları:

LCOM1:0

LCOM2:0

LCOM3:1

LCOM5: 0,0555555555555557

Coh: 0,958333333333333

LCOM4:1

Co:1

TCC: 1 LCC: 1
ROC:0,983613817537644

Yorum: Bazı ölçütler ortak öznitelik kullanımına bağlı olduğu için, bir şekilde ortak kullanılan öznitelik olduğu zaman sınıf uyumu yüksek olarak hesaplanmaktadır. Fakat bu durum tam olarak gerçeği yansıtmamaktadır. Çünkü doğru olmadığı halde sınıf üyeleri tam ilişki içindeymiş gibi bir durum ortaya çıkmaktadır. Dolaylı erişim uyum açısından bir azalmaya sebep olmalıdır fakat doğrudan özellik kullanımına sahip Coh gibi çok yüksek bir düşüşe sebep olmamalıdır. ROC bu durumda sınıfa tam uyum vermemektedir. Beklenen durumda budur.

D11: D2+1 ayrık grup

Kod:

Tablo 5.15: Durum 11 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute4=4;
        }
        public void method2()
        {
```

Tablo 5.15 (Devam): Durum 11 sınıf kodu

```
                attribute2=2;
                attribute3=3;
        }
        public void method3()
        {
                attribute5=5;
                attribute6=6;
        }
        public void method4()
        {
                attribute5=5;
                attribute6=6;
        }
}
```

Uyum Sonuçları:

LCOM1:4

LCOM2:2

LCOM3:2

LCOM5: 0,833333333333333

Coh: 0,375

LCOM4:2

Co:-0,333333333333333

TCC: 0,5 LCC: 0,66

ROC:0,218148148148148

Yorum: Sınıflar içerisinde bazen ayrık gruplar oluşabilmektedir. Ayrık gruplar sınıf içerisinde yer alan üyelerin etkileşimleri açısından ayrık olması anlamına gelmektedir. Bu durum ayrıca tek sorumluluk tasarım prensibine de aykırıdır. Bir küme metot bir küme öznitelik ile etkileşim halindedir. Bu durum kesinlikle istenmeyen bir durumdur. Dikkat edilmediğinde, düşük olması gereken uyumun yüksek elde edilmesine neden olmaktadır. Bu durum bazı ölçütlerde tamamen göz

ardı edilirken bir kısmında ise özniteliklerin eksik kullanımından doğan uyum düşüklüğüne sebep olmaktadır. ROC ölçütü ayrık grupları diğer ölçütlerden farklı olarak özel bir şekilde ele almaktadır.

D12: D2+1 metot tüm özniteliklere erişir, 3 metot bu metodu çağırır

Kod:

Tablo 5.16: Durum 12 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method2()
        {
                method1();
        }
        public void method3()
        {
                method1();
```

Tablo 5.16 (Devam): Durum 12 sınıf kodu

```
        }
        public void method4()
        {
            method1();
        }
    }
```

Uyum Sonuçları:

LCOM1:6

LCOM2:6

LCOM3:4

LCOM5: 1

Coh: 0,25

LCOM4:1

Co:0

TCC: 0 LCC: 0,5

ROC:0,605263157894737

Yorum: Ölçütlerin çoğu, metotların öznitelikleri kullanılması üzerine kurulmuştur. Fakat uyum hesabında bu durum tek başına yeterli değildir. Çünkü metotlar birbirlerini metot çağrımları ile kullanabilirler ve böylece dolaylı yoldan özniteliklere erişebilirler. Bu durumu hesaba katmayan ölçütlerin uyumu doğal olarak düşük çıkmaktadır. ROC ölçütü bu durumu dikkate almakta ve hesaplamaları buna göre yapmaktadır.

D13: D2+1. metot tüm özniteliklere erişir, 2. 1.'ye erişir, 3. 2.'ye erişir, 4. 3.'ye erişir

Kod:

Tablo 5.17: Durum 13 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute4=4;
                attribute5=5;
                attribute6=6;
        }
        public void method2()
        {
                method1();
        }
        public void method3()
        {
                method2();
        }
        public void method4()
        {
                method3();
        }
}
```

Uyum Sonuçları:

LCOM1:6

LCOM2:6

LCOM3:4

LCOM5: 1

Coh: 0,25

LCOM4:1

Co:0

TCC: 0 LCC: 0,5

ROC:0,578282828282828

Yorum: Metotlar doğrudan özniteliğe erişen metotları çağırabileceği gibi, birkaç metot çağrımı ile bu durum gerçekleşebilir. Bu durum uyum hesabında göz önüne alınmalıdır. Özniteliği doğrudan kullanan metodu çağırmak ile birkaç metot çağrımı ile özniteliğe erişmek arasında fark bulunmalıdır. Çünkü ikinci durum, üyeler arasındaki bağın daha zayıf ve kırılgan olduğunu göstermektedir. Sonuçlardan görülebileceği gibi diğer ölçütler bu durumu dikkate almamaktadır. ROC bu durumu dikkate almakta ve göreceli olarak uyum değerinin azalmasına sebep olmaktadır.

D14: D2+2 ayrık grup ve kullanılmayan bir öznitelik

Kod:

Tablo 5.18: Durum 14 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
```

Tablo 5.18 (Devam): Durum 14 sınıf kodu

```
{
        attribute1=1;
        attribute2=2;
        attribute4=4;
    }
    public void method2()
    {
        attribute2=2;
        attribute3=3;
    }
    public void method3()
    {
        attribute5=5;
        attribute6=6;
    }
    public void method4()
    {
    }
}
```

Uyum Sonuçları:

LCOM1:5

LCOM2:4

LCOM3:3

LCOM5: 0,944444444444445

Coh: 0,291666666666667

LCOM4:3

Co:NaN

TCC: 0,33 LCC: 0,33

ROC:0,150320512820513

Yorum: Ayrık gruplar ve kullanılmayan özelliklerin bir arada bulunması uyumu kötü yönde etkileyen bir durumdur. Birçok ölçüt bu durumu, özniteliklerin az kullanılması sonucu uyumun düşmesi ile elde etmektedir. Fakat ROC bu durumu ayrı olarak ele almaktadır. Ayrık grup sayısındaki artış özniteliklerin az kullanımı yanında, ayrık grup etkisi ile düzenlenmektedir.

D15: D4+1 etkileşim yazma durumunda iken okuma durumuna çevrilir

Kod:

Tablo 5.19: Durum 15 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                if(attribute4==4)
                {
                        Console.WriteLine("attrb4 equals 4");
                }
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
```

Tablo 5.19 (Devam): Durum 15 sınıf kodu

```
        public void method3()
        {
                attribute1=1;
                attribute3=3;
                attribute4=4;
                attribute6=6;
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
        }
}
```

Uyum Sonuçları:

LCOM1:1

LCOM2:0

LCOM3:1

LCOM5: 0,722222222222222

Coh: 0,458333333333333

LCOM4:1

Co:0,666666666666667

TCC: 0,66 LCC: 1

ROC:0,602564102564103

Yorum: Metotların öznitelikleri kullanmaları, uyumu belirleme açısından temel teşkil etmektedir. Bu noktada farklı bir durum ortaya çıkmaktadır: Bir metot bir özniteliğe yazma durumu ile erişiyor fakat diğer bir metot aynı özniteliğe okuma durumu ile erişiyor ise bu durumda metot bazında ve sınıf bazında uyum durumu ne olmalıdır? Bu tür okuma ve yazma işlemleri tek yönlü işlemlere göre üyeleri daha fazla bir arada tutmaktadır. Çünkü öznitelikler bir anlamda sınıfın işlevini sunması için işbirliği yapmaktadırlar. Diğer ölçütler okuma ya da yazma durumuna dikkat

etmemektedirler, hâlbuki ROC metot bazında ve sınıf bazında bu durumu uyum belirleme sürecine katmaktadır. Üyeler arasında bir işbirliği olduğu için bu durum uyumu belirli bir miktarda arttırmaktadır.

D16: D4+bağımlı bir öznitelik eklenir

Kod:

Tablo 5.20: Durum 16 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=attribute4+4;
        }
        public void method2()
        {
                attribute2=2;
                attribute6=6;
        }
        public void method3()
        {
                attribute1=1;
                attribute3=3;
                attribute4=4;
                attribute6=6;
```

Tablo 5.20 (Devam): Durum 16 sınıf kodu

```
        }
        public void method4()
        {
                attribute4=4;
                attribute5=5;
        }
}
```

Uyum Sonuçları:

LCOM1:1

LCOM2:0

LCOM3:1

LCOM5: 0,722222222222222

Coh: 0,458333333333333

LCOM4:1

Co:0,666666666666667

TCC: 0,66 LCC: 1

ROC:0,621301775147929

Yorum: Bir sınıfa ait özniteliğin sık kullanılması, bu özniteliği kullanan metotlar arasında sağlam bağlar oluşmasına neden olmaktadır. Bunun yanında sınıf içinde bazen öznitelikler diğer özniteliklerin oluşturulmasında yardımcı olmakta, böylece öznitelikler arasında bağımlılığa sebep olmaktadırlar. Böylece öznitelikler arasında metot ile öznitelik arasındaki ilişkilere benzeyen ilişkiler oluşmaktadır. Öznitelikler arasında ilişkiler oluştuğunda, ROC dışındaki ölçütler bu durumu hesaplama süreçlerine katmamaktadırlar. ROC ölçütü için bu durum bağımlı öznitelikler üzerinden işlemektedir. Bağımlı öznitelik, değeri diğer öznitelikler yardımı ile hesaplanan özniteliklerdir. Bağımlı öznitelik, değerini diğer özniteliklerden elde ettiği için onu kullanan metotlar bir anlamda bağlı olunan öznitelikleri de kullanmış olmaktadırlar. Bu durum üyeler arasındaki bağları sağlamlaştırdığı için uyumu olumlu yönde etkilemektedir.

D17: D2+metotlar ve öznitelikler arasında minimum etkileşim (1)

Kod:

Tablo 5.21: Durum 17 sınıf kodu

```
public class SampleClass
{

        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;

        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
        }

        public void method2()
        {
                attribute2=2;
                attribute4=4;
        }

        public void method3()
        {
                attribute3=3;
                attribute5=5;
        }
```

Tablo 5.21 (Devam): Durum 17 sınıf kodu

```
        public void method4()
        {
                attribute4=4;
                attribute6=6;
        }
}
```

Uyum Sonuçları:

LCOM1:3

LCOM2:0

LCOM3:1

LCOM5: 0,833333333333333

Coh: 0,375

LCOM4:1

Co:0

TCC: 0,5 LCC: 1

ROC:0,488888888888889

Yorum: Sınıf, metotları vasıtası ile özniteliklerin kullanılması ile tam etkileşime sahip olduğunda bazı ölçütler etkileşimin derecesine bakmaksızın uyumu tam olarak hesaplamaktadır. Bazı ölçütler ise etkileşimin sayısına bakarak sınıf içerisinde ayrık grup oluşmuş olsa bile uyum değerini yüksek olarak elde edebilmektedir. ROC metriği böyle bir durumda diğer durumlara nazaran farklı uyum değerini elde edebilmektedir. Bunun sebebi ROC'un fazla parametreye değer vermesinden kaynaklanmaktadır. Böylece her duruma özgü uygun bir uyum değeri elde edilebilmektedir. Ayrıca en az seviyede etkileşime sahip bir sınıfın uyum değerinin 0-1 ölçeğinde ortalarda olması mantıklı bir durum olarak karşımıza çıkmaktadır.

D18: D2+metotlar ve öznitelikler arasında minimum etkileşim (2)

Kod:

Tablo 5.22: Durum 18 sınıf kodu

```
public class SampleClass
{
        private double attribute1;
        private double attribute2;
        private double attribute3;
        private double attribute4;
        private double attribute5;
        private double attribute6;
        public void method1()
        {
                attribute1=1;
                attribute2=2;
                attribute3=3;
                attribute6=6;
        }
        public void method2()
        {
                method1();
        }
        public void method3()
        {
                method4();
        }
        public void method4()
        {
                attribute2=2;
        }
```

Uyum Sonuçları:

LCOM1:5

LCOM2:4

LCOM3:3

LCOM5: 0,944444444444445
Coh: 0,291666666666667
LCOM4:2
Co:0
TCC: 0,16 LCC: 0,5
ROC:0,537142857142857

Yorum: 17. Duruma yapılan açıklama bu madde içinde geçerlidir. Farklı olarak metot işletimleri vasıtası ile tam etkileşim sağlanmaktadır. Metot işletimlerini uyum hesabında kullanmayan ölçütler bu durumda uyumu eksik olarak hesaplamaktadırlar. Hâlbuki metot çağrımları bir metot diğer metotlar üzerinden özelliklere erişebilir, bu durum ise özelliğin doğrudan kullanılmasa bile dolaylı olarak kullanıldığını göstermektedir. Eğer bu duruma dikkat edilmez ise sınıf uyumu, kullanılan metot çağrımlarına bağlı olarak oldukça düşük olabilmektedir. ROC bu durumun üstünden gelebilecek parametrelere sahip olduğundan uyum değerini doğru olarak belirleyebilmektedir.

5.4.2. Ölçüm sonuçları ve değerlendirilmesi

18 test sınıfı için uyum ölçütlerinin hesapladığı değerler Tablo 5.23 içinde verilmiştir.

Tablo 5.23: Ölçüm sonuçları

	LCOM1	LCOM2	LCOM3	LCOM5	Coh	LCOM4	Co	TCC LCC	ROC
1	0	0	0	-	-	0	1	-/-	0
2	6	6	4	1.33333	0	4	1	0/0	0
3	6	6	4	1.27778	0.04167	4	1	0/0	0.004785
4	1	0	1	0.72222	0.45833	1	0.66667	0.5/0.83	0.606061
5	1	0	1	0.83333	0.375	1	0.66667	0.83/1	0.420635
6	0	0	1	0.66667	0.5	1	1	1/1	0.641026
7	2	0	1	0.77778	0.41667	1	0.33333	0.66/0.66	0.518182
8	1	0	1	0.72222	0.45833	1	0.66667	0.66/0.66	0.609023
9	0	0	1	0	1	1	1	1/1	1
10	0	0	1	0.05556	0.95833	1	1	1/1	0.983614
11	4	2	2	0.83333	0.375	2	-	0.5/0.66	0.218148
12	6	6	4	1	0.25	1	0	0/0.5	0.605263
13	6	6	4	1	0.25	1	0	0/0.5	0.578283
14	5	4	3	0.94445	0.29167	3	-	0.33/0.33	0.150321
15	1	0	1	0.72222	0.45833	1	0.66667	0.66/1	0.602564
16	1	0	1	0.72222	0.45833	1	0.66667	0.66/1	0.621302
17	3	1	0	0.83333	0.375	1	0	0.5/0.5	0.488889
18	5	4	3	0.94445	0.29167	2	0	0.17/0.5	0.537143

Gerçekleştirilen ölçütlerin aynı uyum değerini hesapladıkları durumlar Tablo 5.24'de gösterilmektedir. Buna göre ROC haricindeki diğer ölçütler değişen durumlara karşı davranış göstermede eksik kalmaktadırlar. Bu durum ölçütler için sorgulanabilir bir durumdur. Çünkü görülen davranış eksikliği ölçüte olan güvenilirliği azaltmaktadır.

Gerçeklenmiş ölçütlerin uygun uyum değeri üretemedikleri durumlar Tablo 5.25'de verilmektedir. Uygun uyum değeri üretilememesi yapılan değişimlere zamanında tepki verilememesinden daha fazla güven kaybı yaşatan bir durumdur. Çünkü ölçütler ilgili kişilere durumu yönetebilmek için bilgi vermek ile yükümlüdürler. Bu görevin yerine getirilememesi neticesinde sorunlu tasarımlar ortaya çıkacaktır.

Tablo 5.24: Aynı sonuçları üreten sınıf grupları

Ölçüt	**Örnek Sınıflar**
LCOM1	(6,9,10)-(4,5,8,15,16)-(14,18)-(2,3,12,13)
LCOM2	(2,3,12,13)-(4,5,6,7,8,9,10,15,16)-(14,16)
LCOM3	(2,3,12,13)-(4,5,6,7,8,9,10,15,16)-(14,18)
LCOM5	(4,8,15,16)-(5,17,11)-(14,18)-(12,13)
Coh	(4,8,15,16)-(5,17,11)-(14,18)-(12,13)
LCOM4	(2,3)-(4,5,6,7,8,9,10,12,13,15,16,17)-(11,18)
Co	(2,3,6,9,10)-(4,5,8,15,16)-(12,13,17,18)
TCC	(2,3,12,13)-(7,8)-(6,9,10)-(15,16)
LCC	(2,3)-(5,6)-(9,10)-(12,13,17,18)-(15,16)
ROC	-

Tablo 5.25: Uyumu uygun olarak hesaplanamamış sınıflar

Ölçüt	**Örnek Sınıflar**
LCOM1	3.5.8.10.12.13.15.16.17.18
LCOM2	3.5.6.7.8.10.12.13.15.16.17.18
LCOM3	3.5.6.7.8.10.15.16.12.13.17.18
LCOM5	8.12.13.14.17.18
Coh	8.11.12.13.14.15.16.18
LCOM4	3.5.6.7.8.10.12.13.15.16.17
Co	3.5.8.10.11.14.15.16
TCC	2.4.8.11.12.13.14.15.16.18
LCC	2.4.8.10.11.14.15.16
ROC	-

LCOM1-5 normalizasyon ve monoton olma özelliklerini sağlamamaktadır. Bunun yanında, LCOM ölçütleri çoğu durumda uygun olmayan uyum değerlerinin elde edilmesine sebep olmaktadırlar. Özellikle LCOM2 ölçüt değeri birçok sınıf için farklı değer beklenirken 0 değerini almaktadır. LCOM3, LCOM4 ve LCC sınıf üyeleri bir şekilde etkileşimde olduklarında azami uyum değerini verme sorununa sahiptirler (D6, D7). Hâlbuki uyum, değerini etkileşimin yoğunluğuna göre değişebilmelidir. Co, LCOM4'ün sebep olduğu bu kötü durumun üzerinden gelecek düzenlemeye sahip olsa da hala D6 ve D7 durumları için uyum değerini doğru

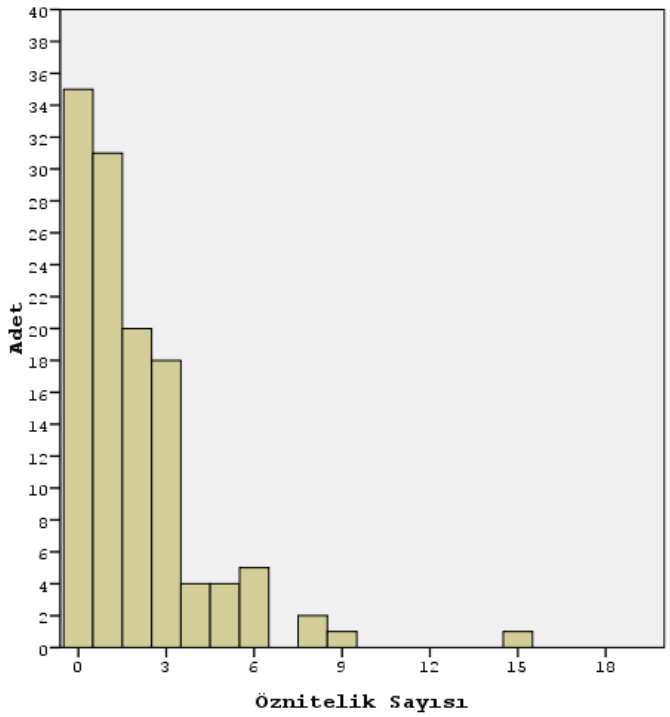
ayarlayamamaktadır. LCOM5 ve Coh farklı bir probleme sahiptirler. Bu ölçütler doğrudan erişilen özniteliklerin sayısını kullanırlar, fakat metot çağrımları ile erişilen öznitelikleri kullanmazlar. Bu ölçütler D10'da olduğu gibi uygun olmayan uyum değerlerini üretirler. Bunların yanında, D9'da olduğu gibi ayrık gruplar ölçütler tarafından değerlendirilmemektedir. D12'de olduğu gibi öznitelikler arasındaki etkileşimler ve D6'da olduğu gibi metotlar ve öznitelikler arasındaki erişim sayıları uyum hesaplama sürecine katılmamaktadırlar. Bunların yanında, özniteliklerin yazılması ve okunması önemli olduğu halde, etkileşimin tipi diğer ölçütler tarafından önemli görülmemektedir (D11).

Yukarıdaki sonuçlara bakıldığında, ROC çeşitli durumlara karşı sınıf uyumunu uygun şekilde hesaplayabilmektedir. Çünkü ROC sınıf karakteristiklerine odaklanan çeşitli kriterleri esas almaktadır. Önerilen ölçütün etkinliği yapılan çalışma ile görülmektedir. Bu sonuç önceden tanımlanmış sınıflar ile gerçekleştirilmiştir. Ölçütün etkinliliğini daha iyi görebilmek için yapılan bir sonraki çalışma açık kaynak kodlu bir yazılım ile deneysel çalışması yapılmasıdır.

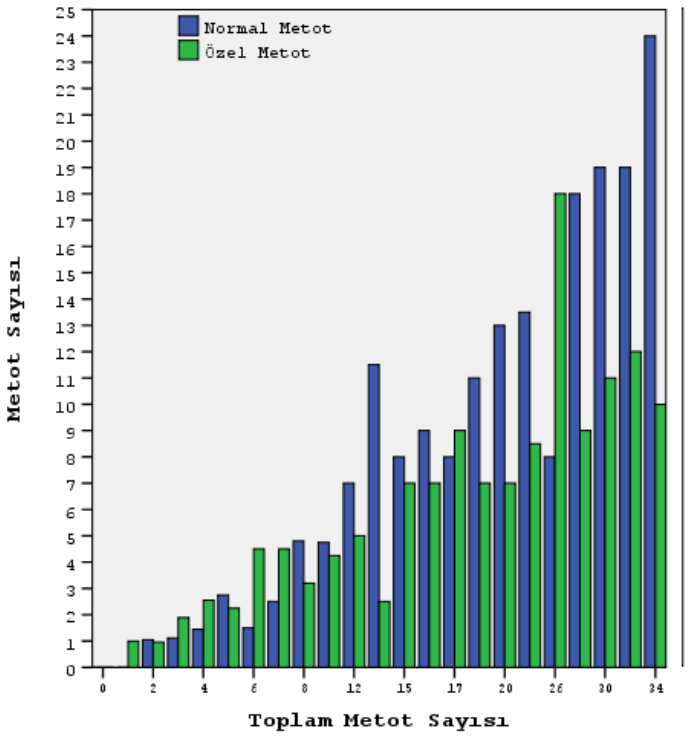
5.5. Deneysel Çalışma 3

Önerilen ölçüt olan ROC ölçütünün etkinliğini göstermek için, The Apache Software Foundation [63] bünyesinde geliştirilmiş olan ve programcı çıktı günlüklerini çeşitli hedeflere yönlendirmede kullanılan log4Net [64] sistemi üzerinde bir deneysel çalışma gerçekleştirilmiştir. Log4Net, programcının hata ayıklama, kullanıcı yönlendirme gibi amaçlarla kullandığı ve günlükleme çıktılarının ekrana veya dosyaya çeşitli formatlarla aktarılmasını sağlayan bir küme sınıf içermektedir.

Log4Net içinde yer alan 159 sınıfın 38 tanesi sanal sınıf veya arayüz olarak işaretlenmiştir. Arayüzler ve sanal sınıflar metotların sadece imzalarını içerdiğinden dolayı bu tür yapılar için uyum hesabı mümkün değildir. Yapılan ölçümler 121 sınıf üzerinden yapılmıştır. Arayüz sınıfları, sanal sınıflar ve sadece bazı yapıları ve sıralama tanımlarını içeren yapılar analiz dışında tutulmuştur. Sınıflara dair öznitelik ve metot dağılım sayıları gerçek sınıflar temel alınarak çıkarılmıştır.

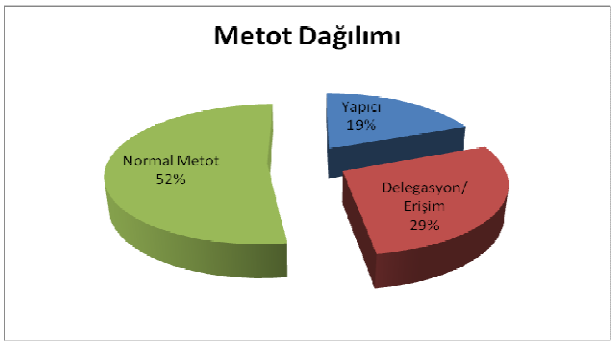


Şekil 5.5: Log4Net sınıflarına ait öznitelik sayılarının dağılımı



Şekil 5.6: Log4Net sınıflarına ait metot (normal ve özel) sayılarının dağılımı

Şekil 5.5 ve Şekil 5.6, çalışmada kullanılan log4Net yazılımında yer alan 121 sınıf içindeki metotların ve özniteliklerin dağılımını göstermektedir. Şekil 5.5'de görüldüğü gibi, birçok sınıf az sayıda öznitelik barındırmaktadır; sınıfların yaklaşık olarak %89'unun dörtten fazla özniteliği bulunmamaktadır. Şekil 5.6 log4Net metotlarını toplam metot sayısına göre normal ve özel metot dağılımı şeklinde göstermektedir. Sınıfların yaklaşık olarak %18'inin 10'dan fazla metodu bulunmaktadır. Metotların bazıları ise 30'dan fazla metot içermektedir. Özel metotların çıkarılması, metot sayısı az olan sınıfların artmasına neden olmaktadır. Özel metotlar çıkarıldıktan sonra yaklaşık olarak sınıfların %13'ü 10 metottan daha fazla metot içermektedir, ayrıca metot sayısı en fazla 20 ile 30 arasında olacak şekilde düşmektedir.



Şekil 5.7: Log4Net Sınıflarına Ait Metotların Türe Göre Sınıflandırılması

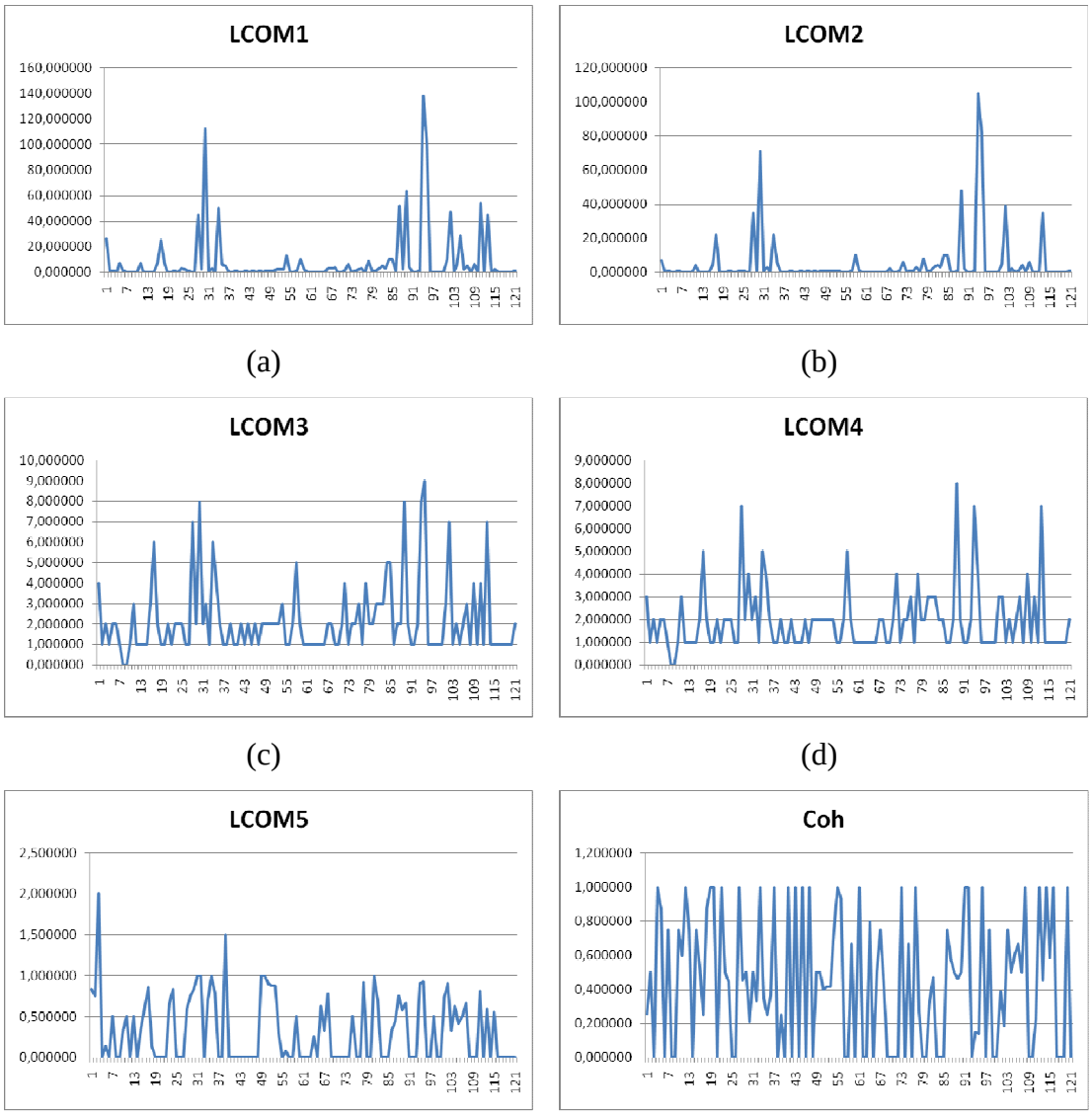
Şekil 5.7 Log4Net içindeki sınıfların metot tür dağılımlarını göstermektedir. Yaklaşık olarak metotların %48'i özel metot olarak belirlenmiştir. Özel metotlar içerisinde yapıcılar, delegasyon ve erişim metotları yer almaktadır. C# diline özgü olan ve delegasyon ile erişim metotlarının birleşimi olan özellik (property) metotları da bu grupta yer almaktadır. Özel metotların oranı göz önüne alındığında, uyumun etkin bir şekilde hesaplanabilmesi için önerilen ölçütte bu metotların ölçüm süreci dışında tutulmasının önemi ortaya çıkmaktadır.

5.5.1. Ölçüm sonuçları ve değerlendirilmesi

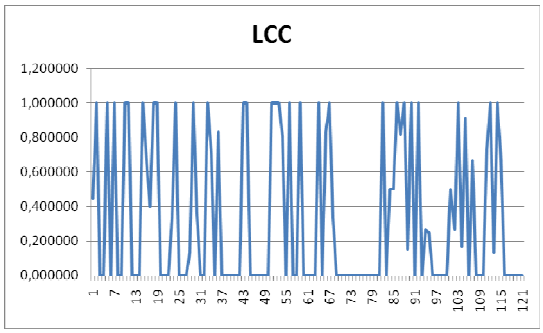
Log4Net uygulaması sınıfları genellikle az sayıda metot ve öznitelikten oluştuğundan dolayı ölçüm sonuçları 0 ya da 1 mertebesinde çıkmıştır. Genellikle uygulamalar ana sınıflar ve yardımcı sınıflar olarak ikiye ayrılmaktadır. Ana sınıflar

belirlenen yazılım mimarisinde genel işlevleri içeren ve yöneten sınıflardır. Yardımcı sınıflar ise belirli fonksiyonelliği sunan ve az sayıda öznitelik ve metoda sahip sınıflardır. Bu tür sınıflar, genelde belirli türden verilerden ve bu verilere ulaşmayı sağlayan özel metotlardan meydana gelmektedir. Seçilen uygulamaya bakıldığında yaklaşık olarak 43 tane sınıf bu yardımcı sınıf kimliğinden farklı davranmaktadır.

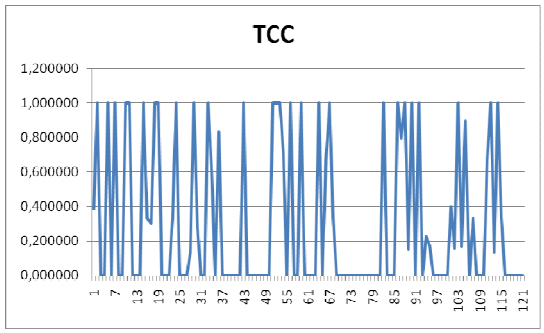
Yapılan çalışma kapsamında 121 sınıfın uyum değeri sonuçları Şekil 5.8'de verilmiştir.



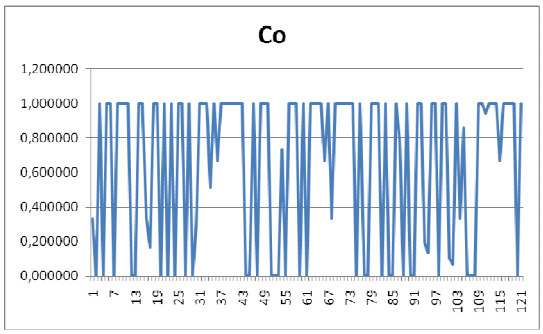
Şekil 5.8: Log4Net Sınıflarının Mevcut Ölçütlere Göre Uyum Değer Grafikleri



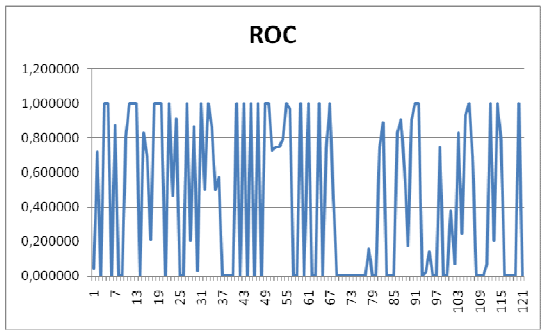
(g)



(h)



(i)



(j)

Şekil 5.8 (Devam): Log4Net Sınıflarının Mevcut Ölçütlere Göre Uyum Değer Grafikleri

Şekil 5.8'de yer alan grafiklerden görüleceği üzere, normalize edilmiş ölçütler arasında ROC daha fazla farklılıkta uyum değeri yakalayabilmektedir. LCOM5 ve Coh ise seçilen yazılım içinde ayrık gruplar ve dolaylı etkileşim çok fazla olmadığından ROC kadar farklılıkta uyum değeri yakalıyor gibi görünseler de hala ayrık grupları ve dolaylı etkileşimleri dikkate almadıkları için her durumda uygun sonuçlar üretememektedirler. ROC ayrıca etkileşim desenine önem verdiğinden ve üyeler arası etkileşimleri tekrar tanımladığından dolayı diğerlerinin maksimum derecede uyumlu gördükleri sınıfları bile farklı derecelerde yüksek uyum değeri ile belirleyebilmektedir.

5.6. Sonuçlar

Yukarıda yapılan deneysel çalışmalar önerilen ölçütün etkinliğini göstermek açısından farklı noktaları bir araya getirmektedirler. Kabul görmüş genel geçer ölçütlerin bulunmadığı ve ölçütlerin etkinliliğini belirlemede yeterli bilgi olmadığı bir durumda tek bir bakış açısı ile ölçütü değerlendirmek yeterli değildir. Bu sebeple bir ölçüt teorik olarak yeteri kadar değerlendirilmeli ve sonrasında deneysel olarak

bu durum desteklenmelidir. Önerilen ölçüt önceki bölümde teorik olarak 2 çatı yardımı ile değerlendirilmiştir. Bu bölümde ise öncelikle hazırlanmış olan 18 adet sınıf ile deneysel olarak değerlendirilmiştir. Bu çalışma şartları belirli bir ortamda ölçütlerin nasıl davrandığını göstermek açısından faydalıdır. Yapılan çalışma ile görülmüştür ki ROC ölçütü değişen durumlara uygun bir şekilde tepki verebilmekte ve diğer ölçütlere göre daha etkin bir çerçeve çizmektedir. Sonrasında yapılan deneysel çalışma ile de geliştirilen ölçütün gerçek ortamda da etkili bir şekilde ölçüm yapabildiği görülmüştür. Geliştirilen ölçütün diğerlerinden farklı bir boyuta sahip olduğunu göstermek için bu aşamadan sonra sonuçlar üzerinde analiz yapılmalıdır. Takip eden bölüm pratik çalışma sonucu elde edilen sonuçların analizini içermektedir.

6. TEMEL BİLEŞEN ANALİZİ

6.1. Giriş

Önceki bölümde yapılan deneysel çalışmaların sonuçlarının değerlendirilmesi bu bölüm kapsamında yapılacaktır. Sonuçlar Principal Component Analysis (Temel Bileşen Analizi) yöntemi ile analiz edilecektir. Kullanılan bu yöntemin amacı ROC ile elde edilen sonuçların mevcut ölçütlerin sonuçlarına göre farklı bir boyuta sahip olup olmadığının tespitidir. Geliştirilen ölçütün teorik ve deneysel olarak değerlendirilmesinin yanında bu tür bir analiz, yapılan çalışmanın mevcut çalışmalardan daha farklı olduğunu göstermek açısından önemlidir. Yapılan analiz ölçütün etkin olduğunu göstermekten ziyade farklı bir bakış açısına sahip olduğunu belirtmektedir. Ölçütün değeri aslında daha önce yapılmış olan teorik ve deneysel değerlendirmeler ile açığa çıkarılmıştır.

6.2. Temel Bileşen Analizi

Sunulan ölçüt mevcut ölçütlere göre farklı karakteristiklere sahiptir. Bu özellikler; ayrık grupların sürece katılması, üyeler arasındaki ilişkilerin yeniden tanımlanması, üyeler arası kullanım sayılarının işlenmesi ve özel metotların süreç dışı tutulmasıdır. Üyeler arası ilişkilerde, kullanım yönü ve niteliği de yazma ve okuma olarak işlenmektedir. Bu noktaları açığa çıkarmak için elde edilen sonuçlar üzerinden Temel Bileşen Analizi [65] metodu ile analiz yapılmıştır. Analiz işlemleri SPSS [66] ve Tanagra [67] programları kullanılarak yerine getirilmiştir.

PCA genellikle veri boyutunu azaltmada (daha az değişken ile ifade edebilme) ve/veya orijinal veri kümesinden yeni ilişkiler çıkarmada kullanılmaktadır. PCA, yüksek miktardaki veri arasından desenler bulmak için genel olarak kullanılan bir tekniktir. Veri içindeki desenlerin belirlenmesinde ve verinin benzerlik ve farklılıkları gösterecek şekilde ifade edilmesinde kullanılan bir yoldur. Yüksek

miktarlardaki veri içindeki desenleri belirlemek ve göstermek zor olduğundan, PCA veri analizinde kullanılan güçlü bir araçtır. PCA'nın diğer bir avantajı ise, yüksek miktardaki veri içindeki desenlerin tespit edilmesinden sonra, verinin daha az boyut ile veri kaybı yaşanmaksızın ifade edilmesini sağlamasıdır [68, 69]. PCA analizi birkaç aşamadan meydana gelmektedir:

1. Verilerin elde edilmesi: Kaç tane sınıfa (boyut) ait kaç tane verinin olduğunun tespit edilmesi işlemidir. Buna göre sınıf kadar satırı ölçüm sayısı kadar sütunu olan bir matris elde edilir.

2. Ortalamanın hesap edilmesi: Her bir boyut için ölçüm verilerinden ortalama elde edildikten sonra, ortalama boyutun her bir veri değerinden çıkarılır. Böylece ortalaması sıfır olan bir veri kümesi elde edilmiş olur.

3. Kovaryans matrisinin hesaplanması: 2. aşamada elde edilen matris üzerinden kovaryans matrisi elde edilir.

4. Özvektörlerin ve özdeğerlerin hesaplanması: 3. aşamada elde edilen kovaryans matrisi üzerinden özvektör ve özdeğer hesabı yapılır. Buna göre ölçüm sayısı kadar özdeğer elde edilir.

5. Bileşenlerin seçilmesi: Elde edilen özdeğerler büyükten küçüğe göre sıralandığında bu özdeğerlere denk düşen özvektörler temel bileşenlerdir (principal component [PC]). Özvektörler veri kümesi içindeki desenler hakkında bilgi verirler. Özvektörler verilerin dağılımlarını karşılayacak en uygun desenleri ortaya çıkarırlar. İlk PC veri boyutları arasındaki en yüksek değişimi ifade eder. Sonrakiler daha az değişimi ifade ederler. Değişimi çok fazla olmayan PC'ler çıkarıldığında veri daha az boyut ile ifade edilmiş olur. Seçilen bileşenler özellik vektörünü meydana getirirler. Daha küçük özdeğer, toplam değişime ağırlık olarak daha az katkıda bulunurlar. Çoğu durumda, ilk birkaç tane bileşen neredeyse tüm değişimi ifade edebilir.

Briand [68] içindeki çalışması ile yapılacak olan deneylerin tekrarlanabilir ve karşılaştırılabilir olması için verileri analiz etmede bir yöntem önermişlerdir. Bu yöntem 3 adım içermektedir: verilerin toplanması, aykırı değerlerin belirlenmesi ve PCA'nın gerçekleştirilmesi. Bu çalışma kapsamında önerilen metoda göre, aşağıdaki işlemler gerçekleştirilmiştir:

1. Uyum ölçümü için verilerin toplanması: Uyumların mevcut ölçütler ve ROC için hesaplanabilmesi için yardımcı bir araç geliştirilmiştir. Araca verilmeden önce sınıflar üzerinde bazı düzenlemeler yapılmıştır. Yardımcı araç vasıtası ile LCOM1-5, Co, Coh, LCC, TCC ve ROC ölçütleri için log4Net sınıflarının uyumları hesaplanmıştır.

2. Aykırı değerlerin tespit edilmesi: Aykırı değerler, örnek veri küme uzayının boş kısımlarında yer alan veri noktalarını ifade etmektedir. Aykırı değerlerin eklenmesi veya çıkarılması analiz sonuçlarını büyük ölçüde etkileyebilmektedir. Bu çalışmada aykırı noktalar tespit edilmiş ve örnek veri kümesinden çıkarılmıştır.

3. Temel bileşen analizinin gerçekleştirilmesi: Bir veri kümesindeki değişkenlerin grubu güçlü olarak ilişkili ise, bu değişkenler muhtemelen objelerin aynı boyutunu ölçmektedirler. PCA, bir veri kümesindeki değişkenler arasındaki ilişkileri belirlemede kullanılan standart bir tekniktir. Principal Component, bağımsız değişkenler arasındaki doğrusal kombinasyonları ifade etmektedir. İlk hesaplanan PC, tüm değişkenler arasındaki doğrusal kombinasyonlardaki en büyük değişim miktarını ifade eder. İkinci ve diğerleri, değişkenler arasındaki doğrusal ilişkileri ifade ederler ve bir önce hesaplanmış olan PC'lere göre dikeydirler. Genellikle, tüm PC'ler içinde sadece bir kısmı büyük katsayılara sahiptirler ve böylece değişime büyük oranda katkı sağlarlar. Bu tür PC'deki katsayılara yükleme adı verilir. PC'ler elde edildikten sonra döndürme işlemi yapılarak değişkenler arasındaki grupların daha belirgin olması sağlanmaktadır.

Bu çalışmada, 121 gerçek sınıf için mevcut ölçütler ve ROC için uyum değerleri hesaplanmıştır. Sonrasında elde edilen değerler üzerinden aykırı değer tespiti yapılmıştır ve 7 tane aykırı değer tespit edilmiştir. Böylece, 114 sınıf üzerinde PCA

analizi yapılmıştır. Tablo 6.1, 114 sınıf için uyum ölçütlerine dair betimleyici istatistikleri göstermektedir. Betimleyici istatistikler, üzerinde çalışılacak verinin basit özelliklerini tanımlamak için kullanılmaktadır. Örnekler ve ölçümler hakkında basit bir özet sunarlar. Betimleyici istatistikler ile verinin neyi gösterdiği basitçe tanımlanmaktadır. Tabloda yer alan dördebölenler (%25 ve %75) ortalamadan ziyade dağılımın yayılımı hakkında bilgi vermektedir.

Tablo 6.2'de Log4Net sınıfları için yapılan ölçümlerin PCA analizine sokulduktan sonra elde edilen sonuçları yer almaktadır. Bu değerler elde edilirken daha temiz bir veri görünümü elde edebilmek için varimax döndürme [70] işlemi yapılmıştır.

Tablo 6.1: Uyum ölçütleri için betimleyici istatistikler

Ölçüt	Min	Maks	25%	Orta	75%	Ortalm	Stan.Sap
LCOM1	0,000000	101,000000	2	3,5	9,75	13,447368	22,45801
LCOM2	0,000000	82,000000	0	1	4	7,710526	16,7621
LCOM3	1,000000	9,000000	1	2	3	2,815789	2,129018
LCOM5	0,080000	1,000000	0,513889	0,666667	0,822917	0,654513	0,224165
Coh	0,135417	0,933333	0,404167	0,5	0,582428	0,502385	0,173371
LCOM4	1,000000	8,000000	1	2	3	2,552632	1,811336
Co	0,000000	1,000000	0	0,422222	1	0,481322	0,437267
TCC	0,000000	1,000000	0,203571	0,690476	1	0,596021	0,398998
LCC	0,000000	1,000000	0,269048	0,81285	1	0,639918	0,389188
ROC	0,141125	0,968304	0,471154	0,746373	0,830769	0,642994	0,258737

6.3. Analiz Sonuçları

PCA analizinin sonuçları Tablo 6.2'de verilmiştir. Altı PC veri kümesindeki değişimin %97'unu kapsamaktadır. Her bir PC için büyük katsayılar kutu içinde verilmiştir. PC tarafından ifade edilen verinin değişimi, toplam değişim ve PC'lere ait özdeğerler de Tablo 6.2'de verilmiştir.

Tablo 6.2: Döndürülmüş bileşenler

Özellik	PC1	PC2	PC3	PC4	PC5	PC6
LCOM1	-0,1301	0,0499	**0,9891**	-0,0397	0,006	-0,0237
LCOM2	**-0,834**	0,3056	0,0995	0,1471	0,2192	-0,1755
LCOM3	**-0,878**	0,2532	0,135	0,051	0,3146	-0,1141
LCOM5	-0,3052	0,1934	-0,0759	0,2056	**0,8995**	-0,0782
Coh	0,3905	-0,1684	-0,101	-0,1101	**-0,8634**	0,2074
LCOM4	**-0,8431**	0,2548	0,0713	0,0513	0,4235	-0,1414
Co	0,1064	0,1412	0,0423	**-0,9635**	-0,1948	0,023
TCC	0,2767	**-0,9228**	-0,0569	0,0852	-0,1439	0,1613
LCC	0,2596	**-0,9075**	-0,03	0,1173	-0,1928	0,2031
ROC	0,2811	-0,4712	-0,0372	-0,038	-0,259	**0,7929**
Özvektör	5,537593	1,639455	1,0608	0,630442	0,598735	0,348247
Oran	27%	22%	10%	10%	20%	8%
Toplam	27%	49%	59%	69%	89%	97%

Elde edilen sonuçlar ışığında PC'ler aşağıdaki gibi yorumlanabilir:

- PC1 (%27): LCOM2, LCOM3 ve LCOM4 ölçütleri ortak öznitelik paylaşan metot çiftlerini esas almaktadırlar. Ayrıca bu ölçütler normalize değildirler. Bu grupta farklı olarak LCOM4 yer almaktadır. LCOM4 öznitelik paylaşımı yanında metot işletimlerini de dikkate almaktadır. Bu durumda metot işletimlerinin ölçütlerin dağılımında etkisinin olmadığı görülmektedir.

- PC2 (%22): TCC ve LCC ölçütleri ortak öznitelik paylaşan metot çiftleri oranını esas almaktadırlar. Ayrıca metot işletimleri vasıtası ile dolaylı olarak öznitelik paylaşımını hesaba katmaktadırlar. Bu iki ölçüt normalize edilmişlerdir. Üst ve alt değerleri bulunmaktadır.

- PC3 (%10): LCOM1 ölçütü sadece ortak öznitelik paylaşan metot çiftlerini esas almaktadır. LCOM2-4'ten farklı olarak sadece ortak öznitelik paylaşmayan metot sayısını temel almaktadır. Diğerleri bu sayıyı doğrudan ya da bir grafik yardımı ile yorumlamaktadırlar.

- PC4 (%10): Co ölçütü ortak öznitelik paylaşan metot çifti oranını kullanmaktadır. Bunun yanında metot işletimlerini de sürece katmaktadır. TCC ve LCC'den farklı olarak dolaylı olmayan etkileşimleri kullanmamaktadır.

- PC5 (%20): LCOM5 ve Coh ölçütleri, öznitelik kullanımını esas almaktadırlar. Metotların eriştikleri öznitelik sayılarını saymaktadırlar. Bu ölçütlerinde üst ve alt değerleri bulunmaktadır.

- PC6 (%8): ROC ölçütü, sınıf üyeleri arasındaki ilişkileri yeniden tanımlamıştır. Buna göre metotların öznitelik kullanımı, metot çağrımları ve özniteliklerin ilişkilerini temel almaktadır. Metotların, özniteliklere kaç defa eriştikleri ya da ne türde eriştikleri uyum sürecinde kullanılmaktadır. Bunların yanında özel metotlar ve üyeler arasındaki ayrık gruplara özel bir durum olarak bakmaktadır.

Analize göre, ROC kendine ait bir boyutu ifade etmektedir ve ROC PC6 içinde en büyük faktörü ifade etmektedir. Bu sonuçlara göre, sunulan ölçüt sınıfların özelliklerine farklı bir bakış açısı veya yenilik getirmektedir. Sunulan ölçüt getirdiği farklılık ile mevcut olan diğer ölçütlerden ayrılmaktadır. Bu çalışmada yapılan PCA analizi birkaç unsur haricinde diğer çalışmalarda yapılmış olan PCA analizleri ile benzerlik göstermektedir [59, 71, 45]. Tablo 6.3 bahsi geçen çalışmalarda elde edilen sonuçları göstermektedir.

Tablo 6.3: PC sonuçlarının karşılaştırılması

Kendi Sonuçlarımız	**Letha et al. [45]**	**Briand et al. [71]**	**Chae et al. [59]**
PC1 (%27): LCOM2, LCOM3, LCOM4 PC3 (%10): LCOM1	PC1 (%50): LCOM1, LCOM2, LCOM3, LCOM4	PC1 (%49): LCOM1, LCOM2, LCOM3, LCOM4	PC2 (%27,5): LCOM1, LCOM2, LCOM3 PC4 (%5,1): LCOM4
PC2 (%22): TCC, LCC PC4 (%10): Co	PC2 (%33,8): LCOM5, Coh, LCC, TCC	PC2 (%26): Co, LCC, TCC	PC1 (%48,2): LCC, TCC, Co
	PC3 (%8,7): LCOM		
	PC4 (%48): PLCOM1, PLCOM2		
PC5 (%20): LCOM5, Coh		PC4 (%8): LCOM5, Coh	PC3 (%7,9): LCOM5, Coh
PC6 (%8): ROC			
		PC3 (%10): ICH	
			PC3 (%5,9): CBMC

Bu bölüm geliştirilen ölçütün diğer mevcut ölçütlerden farklı bir boyuta sahip olduğunu açıkça ortaya koymaktadır. Temel bileşen analizi ile sonuçlarını analiz eden diğer çalışmaların sonuçları ile kendi sonuçlarımızın paralel çıkması yaptığımız analizin doğru şekilde yapıldığının göstergesidir. Bu açıdan bakıldığında ROC farklı boyutta sonuçlar üretebilen bir ölçüttür. Analiz bir ölçütün doğruluğunu göstermez fakat yapılan çalışmanın tekil olduğunu göstermesi açısından önem taşımaktadır. Ayrıca önceki bölümlerde yapılan teorik ve deneysel çalışmalar ölçütün diğer ölçütler karşısında daha avantajlı olduğunu göstermektedir.

7. SONUÇLAR VE ÖNERİLER

Nesneye yönelik sistemlerin en temel yapı birimi olan sınıf, bazı karakteristiklere sahiptir ve eğer bu tür karakteristikler (metotlar, öznitelikler, bu üyeler arasındaki ilişkiler vb.) üretilecek olan ölçütler için temel oluşturmaz ise elde edilen sonuç sorgulanabilir durumda olacak, dolayısıyla ölçüt sınıf uyumunu tam olarak yansıtamayacaktır.

Sınıf karakteristiklerinin yanı sıra yazılıma yön veren bazı tasarım kalıplarının da ölçütler tarafından karşılanabilir olması gerekmektedir. Tek sorumluluk prensibi gibi bir tasarım belirleme prensibine uygun ölçütler geliştirilmelidir. Bilgi sahibi prensibine göre sınıfa üye eklendiğinde, ölçütün gerçekten bu prensip uygulandığında sonucun mantıklı olup olmadığını ölçüm sonuçları ile ilgili kişilere aktarması gerekmektedir.

Ölçütler temel sınıf karakteristiklerini ve bazı tasarım oluşturma prensiplerini temel almalarının yanında, ölçü geliştirme kurallarına tam olarak uymaları gerekmektedir. Zira oluşturulan ölçüt, temel ölçüt geliştirme kurallarına uymadığı takdirde elde edilen sonuçların değerlendirilmesi mümkün olamayacaktır. Ayrıca ölçüt geliştirme için bugüne kadar yapılmış ve kabul görmüş prensipler izlenmeden yapılan çalışmalar her zaman sorgulanabilir durumda olacaktır. Geliştirilecek ölçüt bu prensipleri sağladığında bir ölçüt olarak değerlendirilebilecektir.

Yukarıdaki açıklamalardan ve tezin genelinden anlaşılacağı gibi ölçüt geliştirme, ölçülecek alanın iyice anlaşılması yanında, ölçme kavramının iyi anlaşılmasına ihtiyaç duymaktadır. Elde edilen sonucun ise genel geçer mevcut kurallar ile geçerlenmesi gerekmektedir. Ölçütün kabulü ise daha uzun bir sorgulama zamanına ihtiyaç duyacaktır.

Kabul görmüş sınıf uyum ölçütleri, özellikle öznitelik kullanım esası tabanlıdırlar. Bu kıstas önemlidir fakat kesinlikle yeterli değildir ve sınıf içindeki üyelerin ilişkilerini tümüyle değerlendirememektedir. Bu çalışmada, sınıfların karakteristiklerini daha iyi yakalayarak uyum sürecine uygun bir şekilde katabilmek için bir ölçüt geliştirilmiştir. Geliştirilen ölçüt, ROC, özellikle metotlar ve öznitelikler arasındaki farklı etkileşimlere odaklanmaktadır. Bu etkileşimler, metotlar arası çağrım ilişkileri, metotlar ve öznitelikler arasındaki okuma-yazma-erişim ilişkileri ve öznitelikler arası bağımlılık ilişkileridir. Bunların yanında, ölçüt ayrıca etkileşim deseni ve üyeler arasındaki erişim sayılarını sürece katmaktadır. Literatürde bu türde değişik ilişkileri bir arada kullanan bir ölçüt bulunmamaktadır.

Günümüzde gelişen yazılım sistemleri, daha hassas ölçümlerin yapılabilmesine ihtiyaç duymaktadır. Sınıf içi bağımlılıktaki bir değişimin ilgili kişilere uygun şekilde aktarılabilmesi önemlidir. Ayrıca sınıf içi bağımlılıkların değişen durumlar için değişebildiğinin tespiti bir ölçütü güvenilir kılmaktadır. ROC, değişen durumları hassas bir şekilde elde edebildiği gibi, gerekli olduğunda matematiksel modelinin çeşitli evrelerinde verilen çıktılar kontrol edilerek metotların ve özniteliklerin bağımlılıkları ve uyuma katkıları da belirlenebilir. ROC sonuç olarak sınıf uyum ölçütü olsa da modelinin bir önceki çıktıları diğer aşamalara girdi olmadan toplanırsa uyumun metotlar ve öznitelikler açısından nelere değer olduğunu ortaya koyabilmektedir. Bu özellikleri ile mevcut ölçütlerden ayrılmaktadır. Örneğin dört metotlu bir sınıf için, metotların uyuma olan katkıları tek tek alınabilir. Eski yöntemlerde tam uyum (normalize edilmiş 1 değeri) olduğunda tüm metotlar için uyum değeri 0.25 olarak elde edilirken, ROC bu durumu metotların katkılarını gösterecek şekilde yansıtabilmektedir. Örneğin metotlar 0.20, 0.45, 0.10 ve 0.25 uyumunu sergileyebilirler.

Tez kapsamında mevcut ölçütler ve geliştirdiğimiz ölçütün uyum hesaplama süreçlerini kolaylaştırmak için bir yazılım hazırlanmıştır. Deneysel çalışmalar için gerekli olan sonuçların üretilmesi bu yazılım yardımı ile sağlanmıştır. Oluşturulan yazılım bir ürün olmamakla birlikte ileriye yönelik geliştirilebilir bir durumdadır. Bir test ortamı sağlayabileceği gibi ileride kurumsal kullanım için uygun bir duruma getirebilir niteliktedir.

ROC ölçütünün etkinliğini göstermek için, önceden hazırlanmış özel sınıflar üzerinde ve Log4Net sınıfları üzerinde deneysel çalışmalar yapılmıştır. Elde edilen sonuçlar ROC'un etkinliğini ve verimliliğini açık olarak ortaya koymaktadır. Buna göre, ROC, mevcut ölçütlerin kısıtlarının üstesinden gelebilmektedir ve sınıfların uyum değerlerini objektif bir şekilde değerlendirebilmektedir. Önceden hazırlanmış sınıflar üzerinde yapılan deneysel çalışma, kontrollü bir test ortamında geliştirilen ölçütlerin nasıl davrandığını göstermek üzerine yapılmıştır. Açık kaynak kodlu yazılımlar üzerinde yapılan deneysel çalışmalar, sonuç olarak bir temizleme işlemine gerek duymaktadır ve bu durum yapılan çalışmanın niteliğine göre değişiklik gösterebilmektedir. Hâlbuki önceden hazırlanmış kontrollü sınıflar ölçütlerin değerlendirmeleri açısından bir referans oluşturabilmektedir. Bu tür bir çalışma genel olarak literatürde yer almamaktadır. Bu çalışmaya özgü sonuçlardan biri de bu tür bir kontrollü çalışmanın yapılmış olmasıdır.

Deneysel çalışmalar, ROC'un diğer ölçütlere göre daha fazla farklılıkta ve hassaslıkta uyum değeri yakalayabildiğini göstermektedir. Çünkü ROC üyeler arasındaki ilişkileri yeniden tanımlamaktadır. Bunların yanında, mevcut ölçütler ile önerilen ölçüt arasındaki ilişkileri belirleyebilmek ve ROC'un farklılığını tespit edebilmek için Log4Net sınıfları üzerinde temel bileşen analizi uygulanmıştır. Temel bileşen analizi elde edilen sonuçların farklı bir boyuta sahip olup olmadığını göstermek için kullanılan yaygın bir analiz yöntemidir. Doğruluktan ziyade elde edilen verilerin diğer veri gruplarından farklı bir boyutu ifade ettiğini gösterebilmektedir. Analiz sonuçları, mevcut ölçütler tarafından yakalanamayan sınıf özelliklerinin ROC tarafından yakalandığını göstermiştir. Bu sonuç ROC'un mevcut ölçütlerden farklı bir yaklaşıma sahip olduğunu göstermek açısından çok önemlidir. Ayrıca yapılan bu çalışma literatürde yer alan benzer çalışmalarla uyum göstermektedir. Bu durumun önemi yapılan analizin doğrulanması açısından önemlidir.

Tez kapsamında tanımlanan ilişkilerin bir kısmı sadece bu çalışmaya özeldir. Mevcut ölçütlerde olduğu gibi ROC'da temelde metotların öznitelik kullanımına bağlıdır. ROC yapısal bir uyum ölçütüdür. Bunun yanında metotların öznitelikleri kaç defa

kullandıkları ve kullanımın yazma ya da okuma olması, metot çağrımlarındaki ağırlıkların dağıtılması ve öznitelikler arasındaki ilişkilerin ağırlıklandırılması sadece bu çalışmaya özeldir. Ayrık grupların sürece katılması da sadece bu çalışmaya özeldir. Tüm bu ilişkilerin yerinde tanımlanması önemlidir. Bu tanımlamalar uygun şekilde yapıldığında çalışma çeşitli şekillerde genişletilebilir. Böylelikle zamanımızda karmaşıklaşan sistemler için daha uygun ve hassas ölçümler yapabilen ölçütler geliştirilebilir.

Gelecek çalışması, geliştirilen ölçütün, nesneye yönelik tasarımın kalıtım, polimorfik metotlar gibi diğer önemli konuları için daha ileriye götürülmesi olacaktır. Ayrıca sınıf üyeleri arasındaki farklı türde ilişkilerin keşfedilmesi ve ilişkilerin etkin bir şekilde sürece katılmaları üzerinde çalışmalar yapılacaktır. Bu tür detaylar, sınıfın karakteristiklerini etkin olarak kullanan programlar için uyumun tespitinde önem kazanmaktadır. Çünkü bahsedilen tüm ilişkiler aslında sınıfların karakteristik özellikleridir ve kesinlikle uyum hesaplama sürecinde hesaba katılmalıdırlar. Önemli bir diğer konu ise, uyuma katılması muhtemel kıstasların yeteri kadar ayırt edici olup olmadıklarının tespiti sorunudur. Bunun yanında sınıfların durumuna göre gerekli kıstasların dinamik olarak hesaplama sürece katılmaları ileride yapılacak çalışmalar arasında önemli bir yer arz etmektedir.

Günümüze kadar sınıf uyumu konusunda yapılan çalışmaların birbirleri ile kıyas edilmesi her zaman kişisel yorumlara açık durumda kalmıştır. Bu sebepten dolayı, genel geçer ölçütlerin ortaya çıkması mümkün olmamıştır. Fakat bu konuda ilk zamanlarda sunulan ölçütler veya üzerinde değerlendirme yapılan bazı ölçütler literatürde kalıcı hale gelmişlerdir. Bu konudaki sorun, oluşturulan bu ölçütlerin kontrollü örnekler üzerinde denenmemiş olmalarıdır. Çalışmalar genelde açık kaynak kodlu yazılımlar üzerinde yapılmıştır. Kullanılan bu yazılımlar birer temizleme işlemlerinden geçirildikten sonra kullanılmışlardır. Bu temizleme işlemleri, çalışmaları yapılan kişiler tarafından kişisel olarak belirlenmiştir. Temizleme işleminin sübjektif olmasının yanında kullanılmak için seçilen yazılımın tipi de çalışmayı temelden etkilemektedir. Çünkü eğer seçilen yazılım ölçütün dikkat etmediği özelliklere sahip ise bunlar değerlendirilmediği için ölçüm sonuçları olumlu gibi görülebilir. Hâlbuki dikkat edilmeyen özellikler kötü uyum işareti olabilirler.

Tüm bu durumların çözümü önceden hazırlanmış sınıflar üzerinde geliştirilen ölçütlerin sonuçlarının değerlendirilmesidir. Bu tez kapsamında oluşturulan 18 örnek sınıf bu açıdan oluşturulmuştur. İleriye yönelik kontrollü sınıf örnek kümesi oluşturma çalışmasının ilk adımını oluşturmaktadır. Bu tür bir küme geliştirilen ölçütler tarafından kullanılabilir. Böylelikle çıkan sonuçlar literatürde yayınlanarak çalışma yapan kişilere kendi ölçütlerini değerlendirmeleri açısından karşılaştırma imkânı sağlamış olur.

Tez kapsamında geliştirilen ROC ölçütü daha önce sempozyumlarda sunulmuş olan COC ölçütü [32, 33] ve onun dayandığı temel noktaları baz almaktadır. COC ölçütü basit olarak sınıf uyumunun en basit şekilde elde edilmesini sağlamak için geliştirilmiştir. ROC ölçütü kapsamında bahsedilen ilişkiler sadece metot-öznitelik ilişkisini içerecek şekilde uygulandığında COC ölçütüne benzer hale gelmektedir. Bu açıdan bakıldığında ROC ölçütü, sınıf uyumu hakkında eğitim verme amacı ile basitliği sayesinde ile kullanılabilir. Bu fayda yapılan çalışmanın sağladığı getirilerden bir diğeri olarak karşımıza çıkmaktadır.

İleride yapılabilecek bir diğer çalışma ise; birçok yazılım üzerinde deneysel çalışmaların yapılarak bir veritabanının oluşturulması olacaktır. Oluşturulan bu veritabanı üzerinde yapılan çalışmalar ile oluşturulacak sonraki yazılımlarda tasarım kalıbı önerisi yapılabilecektir. Örneğin, ayrık grupların fazla olması durumunda sınıfın parçalanması önerisi kolaylıkla sunulabilecektir. Farklı bir çalışma ise sınıfı bir yerleşim yeri olarak ve ilişkileri ise bir veri kümesi olarak görüp SOM (Self Organizing Map [Kendi Kendini Düzenleyen Harita])'lar yardımı ile uyumun görselleştirmesi olacaktır. Bu konularda yapılan çalışma yok denecek kadar azdır. Fakat uygun bir ölçüt geliştirildiği takdirde veya üyeler arası ilişkiler uygun şekilde belirlenebildiğinde bu çalışmaların yapılabilmesi için engel bulunmamaktadır.

8. KİŞİSEL YAYINLAR VE ESERLER

1. Kurubaş, Ö., Duru, N., "Nesneye Yönelik Sınıflar İçin Uyumun Ölçülmesine Dair Akıllı Bir Yaklaşım", ***Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyumu***, ASYU 2008, Haziran (2008).

2. Kurubaş, Ö., Duru, N., "Ayrık Etkileşim Gruplarını Kullanan Yeni Bir Sınıf Uyum Belirleme Önerisi", ***2. Ulusal Yazılım Mimarileri Konferansı***, UYMK 2008, 135-144, (2008).

3. Kurubaş, Ö., Duru, N., "Nesneye Yönelik Sistemler İçin Yeni Bir Sınıf Uyum Kriteri: Etkileşim Gücü, Ayrık Etkileşim Grupları ve Özel Metotlar", ***Bilimde Modern Yöntemler Sempozyumu***, BUMAT 2008, (2008).

4. Kurubaş, Ö., Duru, N., "A Novel Approach About Cohesion Measurement for Classes", ***International Symposium On Computer And Information Sciences***, 23, 123-128, (2008).

5. Kurubaş, Ö., Duru, N., "Sınıf Uyumunu Etkileyen Faktörler ve Bu Faktörlere Uygun Bir Uyum Ölçütünün Geliştirilmesi", ***Yazilim Gelistirme Araçlari Sempozyumu***, YKGS 2011, (2011).

KAYNAKLAR

[1] Pressman, R.S., "Software Engineering, A Practitioner's Approach", ***McGraw-Hill***, United States of America, (2001).

[2] Basili, V., Briand L., Melo, W.L., "A Validation of Object-Oriented Design Metrics as Quality Indicators", ***IEEE Transactions on Software Engineering***, 22, 751-761, (1996).

[3] Li, W., Henry, S., "Object-Oriented Metrics That Predict Maintainability", ***Journal of Systems and Software***, 23, 111-122, (1993).

[4] Bieman, J.M., Kang, B.K., "Cohesion And Reuse In An Object-Oriented System", ***Proceedings of the Symposium on Software Reusability (SSR'95)***, 259-262, (1995).

[5] Briand, L.C., Daly, J., Würst, J., "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", ***Empirical Software Engineering***, 3 (1), 67-117, (1998).

[6] Allen, E., Khoshgoftaar, T., "Measuring coupling and cohesion: an information theory approach", ***Proceedings of the 6. Software Metrics Symposium***, 119-127, (1999).

[7] Fenton, N.E., Pfleeger, S.L., "Software Metrics: A Rigorous and Practical Approach", ***PWS Publishing Company***, (1998).

[8] Yourdon, E., Constantine, L., "Structured Design", ***Prentice Hall***, (1979).

[9] Bieman, J., Ott, L., "Measuring Functional Cohesion", ***IEEE Trans. Software Engineering***, 20(8), 644-657, (1994).

[10] Bieman, J.M., Kang, B.K., "Cohesion and Reuse in An Object-Oriented System", ***ACM Press New York***, 259-262, (1995).

[11] Rumbaugh et al., "Object-Oriented Modeling and Design", ***Prentice Hall***, (1991).

[12] Booch, G., "Object-Oriented Analysis and Design with Applications", ***Addison-Wesley***, (2007).

[13] El Emam, K., Melo, W., "The Prediction of Faulty Class Using Object-Oriented Design Metrics", ***National Research Council of Canada NRC/ERB 1064***, (1999).

[14] Chidamber, S.R., Kemerer, C.F., "Towards a Metrics Suite for Object-Oriented Design, Object-Oriented Programming Systems", ***Languages and Applications (OOPSLA), Special Issue of SIGPLAN Notices***, 26(10), 197-211, (1991).

[15] Henderson-Sellers, B., "Object-Oriented Metrics Measures of Complexity", ***Prentice-Hall***, (1996).

[16] Li, W., Henry, S., "Object Oriented Metrics That Predict Maintainability", ***Journal of Systems and Software***, 23, 111-122, (1993).

[17] Hitz, M., Montazeri, B., "Measuring Coupling and Cohesion in Object Oriented Systems", ***Proceedings of the Int. Symposium on Applied Corporate Computing***, 25-27, (1995).

[18] Chae, H.S., Kwon, Y.R., Bae, D. H., "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables", ***IEEE Transaction on Software Engineering***, *30*, 826-832, (2004).

[19] Zhou, Y., Xu, B., Zhao, J., Yang, H., "ICBMC: an improved cohesion measure for classes", ***Proceedings of International Conference on Software Maintenance***, 44-53, (2002).

[20] Aman, H., Yamasaki, K., Yamada, H., Noda, M-T., "A Proposal Of Class Cohesion Metrics Using Sizes Of Cohesive Parts", ***Knowledge-Based Software Engineering***, 102-107, (2002).

[21] Chen, Z.-Q., Xu, B.-W., Zhou, Y.-M., "Measuring class cohesion based on dependence analysis", ***Journal of Computer Science and Technology***, 19(6), 859-866, (2004).

[22] Marcus, A. Poshyvanyk, D., "The Conceptual Cohesion of Classes", ***In Proceedings, 21st IEEE International Conference on Software Maintenance (ICSM'05)***, 133-142, (2005).

[23] Liu, Y., Poshyvanyk D., Ferenc, R., Gyimothy, T., Chrisochoides, N., "Modeling class cohesion as mixtures of latent topics", ***IEEE International Conference on Software Maintenance***, 233-242, (2009).

[24] Lanza, M. Marinescu, R., "Object-Oriented Metrics in Practice", ***Springer***, (2006).

[25] Bansiya, J., Etzkorn, L., Davis, C., Li, W., "A class cohesion metric for object oriented designs", ***Journal of Object-Oriented Program***, 11(8), 47-52, (1999).

[26] Wang, J., Zhou Y., Wen, L., Chen, Y., Lu, H., Xu, B., "DMC: a more precise cohesion measure for classes", ***Information and Software Technology***, 47(3), 167-180, (2005).

[27] Fernandez, L., Pena, R., "A sensitive metric of class cohesion", ***International Journal of Information Theories and Applications***, 13(1), 82-91, (2006).

[28] Badri, L. Badri, M., "A Proposal of a New Class Cohesion Criterion: An Empirical Study", ***Journal of Object Technology, Special issue: TOOLS 2003***, 3(4), 145-159, (2004).

[29] Dallal, Al. J., Briand, L., "An object oriented high-level design-based class cohesion metric", ***Information and Software Technology 52(12)***, 1346-1361, (2010).

[30] Dallal, Al, J., "Mathematical validation of object-oriented class cohesion metrics", ***International Journal of Computers 4(2)***, 45-52, (2010).

[31] Dallal, Al, J., "Measuring the discriminative power of object-oriented class cohesion metrics", ***IEEE Transactions on Software Engineering***, 1-1, (2010).

[32] Kurubas, O., Duru, N., "A novel approach about cohesion measurement for classes", ***Computer and Information Sciences, 2008. ISCIS '08. 23rd International Symposium***, 1-6, (2008).

[33] Kurubaş, Ö., Duru, N., "Sınıf uyumunu etkileyen faktörler ve bu faktörlere uygun bir uyum ölçütünün geliştirilmesi", ***2. Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu***, (2010).

[34] Beyer, D., Lewerentz, C., Simon, F., "Impact of Inheritance on Metrics for Size, Coupling, and Cohesion in Object-Oriented Systems", ***In Proceedings of IWSM'2000***, 1-17, 2006.

[35] Counsell, S., Swift, S., Crampton J., "The interpretation and utility of three cohesion metrics for object-oriented design", ***ACM Transactions on Software Engineering and Methodology (TOSEM)***, 15(2), 123-149, (2006).

[36] Counsell, S., Mendes, E., Swift S., "Comprehension of object-oriented software cohesion: the empirical quagmire", ***Proceedings of the 10th International Workshop on Program Comprehension (IWPC 2002)***, 33-42, (2002).

[37] Bansiya, J. Davis, C. G., "A hierarchical model for object-oriented design quality assessment", ***IEEE Transactions on Software Engineering***, 28(1), 4-17, (2002).

[38] Briand, L.C., Wüst, J., Daly, J.W., Porter, V.D., "Exploring the relationship between design measures and software quality in object-oriented systems", ***Journal of System and Software***, 51(3), 245-273, (2000).

[39] El-Emam, K., "Object-Oriented Metrics: A Review of Theory and Practice", ***In Advances in Software Engineering***, 23-50, (2002).

[40] Quah, T.-S., Thwin, M.M.T., “Application of neural networks for software quality prediction using object-oriented metrics”, ***In Proceedings of International Conference on Software Maintenance***, 116-125, (2003).

[41] Brito e Abreu, F., Goulao, M., “Coupling and cohesion as modularization drivers: are we being overpersuaded?”, ***In Proceedings of 5th European Conference on Software Maintenance and Reengineering***, 47-57, (2001).

[42] Maletic, J.I., Marcus, A., “Supporting Program Comprehension Using Semantic and Structural Information”, ***In Proceedings of 23rd International Conference on Software Engineering***, 103-112, (2001).

[43] Etzkorn, L.H., Davis, C.G., “Automatically Identifying Reusable OO Legacy Code”, ***IEEE Computer***, 30(10), 66-72, (1997).

[44] Lee, J.K., Jung, S.J., Kim, S.D., Jang, W.H., Ham, D.H., “Component identification method with coupling and cohesion”, ***In Proceedings of Eighth Asia-Pacific Software Engineering Conference***, 79-86, (2001).

[45] Etzkorn, L.H., Gholson, S.E., Fortune, J.L., Stein, C.E., Utley, D., Farrington, P.A., Cox, G.W., “A comparison of cohesion metrics for object-oriented systems”, ***Information and Software Technology***, 46(10), 677-687, (2004).

[46] Chidamber, S.R., Kemerer, C.F., “A Metrics suite for object Oriented Design”, ***IEEE Transactions on Software Engineering***, 20, 476-493, (1994).

[47] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., “Object-Oriented Modeling and Design”, ***Prentice Hall, Englewood Cliffs***, (1991).

[48] Wegner, P., “Dimensions of Object-Based Language Design”, ***Object-Oriented Programming Systems Language and Applications (OOPSLA)***, 22, 168-182, (1987).

[49] Meyer, B., “Object-Oriented Software Construction”, ***Prentice Hall***, (1997).

[50] Meyer, B., “A Overloading vs Object Technology”, ***Journal of Object-Oriented Programming (JOOP)***, 14(4), 3-7, (1989).

[51] Meyer, B., “Touch of Class: Learning to Program Well with Objects and Contracts”, ***Springer***, (2009).

[52] Nierstrasz, O.M., “A Survey of Object-Oriented Concepts”, ***Object-Oriented Concepts, Databases and Applications***, 3-21, (1989).

[53] DeMarco, T., “Structured Analysis and System Specification”, ***Yourdon Press Computing Series***, (1979).

[54] Page-Jones, M., “The Practical Guide to Structured System Design”, ***Yourdon Press Computing Series***, (1988).

[55] IEEE Std 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology”. http://ieeexplore.ieee.org/servlet/opac?punumber=2238, , ***(Ziyaret tarihi: 05.10.2006)***.

[56] Demarco, T., “Controlling Software Projects: Management, Measurement & Estimation”, ***Yourdon Press***, (1982).

[57] Bär, H., Bauer, M., Ciupke, O., Demeyer, S., Ducasse, S., Lanza, M., Marinescu, R., Nebbe, R., Nierstrasz, O., Przybilski, M., Richner, T., Rieger, M., Riva, C., Sassen, A., Schulz, B., Steyaert, P., Tichelaar, S., Weisbrod, J, “The FAMOOS Object-Oriented Reengineering Handbook”, ***http://scg.unibe.ch/archive/famoos/handbook/4handbook.pdf***, Ekim 1999.

[58] Fenton, N.. “Software Measurement : A Necessary Scientific Basis” , ***IEEE Transactions on Software Engineering***, 20(3), 199-206, (1994).

[59] Chae, H.S., Kwon, Y.R., Bae, D.H., “A Cohesion Measure For Object-Oriented Classes”, ***Software Practice And Experience***, 30, 1405-1431, (2000).

[60] Hitz, M., Montazeri, B., “Chidamber and Kemerer’s metrics suite: a measurement theory perspective”, ***IEEE Transactions on Software Engineering***, 22, 267-271, (1996).

[61] Briand, L.C., Morasca, S., Basili, V.R., “Property-based Software Engineering Measurement”, ***IEEE Transaction on Software Engineering 22(1)***, 68-86, (1996).

[62] Kitchenham, S., Pfleeger, S., Fenton, N., “Towards a framework for software measurement validation”, ***IEEE Transactions on Software Engineering 21(12)***, 929-944, (1995).

[63] http://www.apache.org/, ***(Ziyaret tarihi: 20.04.2010)***.

[64] http://logging.apache.org/log4net/release/features.html, ***(Ziyaret tarihi: 20.04.2010)***.

[65] Dunteman, G., “Principal Component Analysis”, ***SAGE Publications***, (1989).

[66] http://www.spss.com.tr/, ***(Ziyaret tarihi: 11.10.2010)***.

[67] http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html, ***(Ziyaret tarihi: 15.10.2010)***.

[68] Briand, L., Würst, J., Daly, J., Porter, V., “Exploring The Relationships Between Design Measures And Software Quality In Object-Oriented Systems”, ***Journal of Systems and Software No. 51***, 245-273, (2000).

[69] Dillion, W.R., Goldstein, M., “Multivariate Analysis: Methods and Applications”, ***Wiley***, (1984).

[70] Jolliffe, I.T., “Principal Component Analysis”, ***Springer Verlag***, (1986).

[71] Briand, L.C., Daly, J., Porter, V., Würst J., “A comprehensive empirical validation of design measures for object-oriented systems”, ***Proceeding of the Fifth International Software Metrics Symposium***, 246-257, (1998).

ÖZGEÇMİŞ

1979 yılında Bursa'da doğdu. İlk, orta ve lise öğrenimini Bursa' da tamamladı. 1998 yılında birinci olarak girdiği Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2002 yılında bölüm birincisi olarak mezun oldu. 2002-2005 yılları arasında Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Öğrenimini tamamladı. 2005 yılında Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda Doktora programına başladı. 2002-2008 yılları arasında Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak görev yaptı. 2008 yılında TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü' de Uzman Araştırmacı olarak görev yapmaya başladı. 2008 yılından beri aynı enstitüde Uzman Araştırmacı-Yazılım Süreç Sorumlusu olarak görevine devam etmektedir.