YILDIZ TEKNİK ÜNİVERSİTESİ

BİLİMSEL ARAŞTIRMA PROJELERİ KOORDİNATÖRLÜĞÜ

BİTİRME RAPORU

Proje No: 28-04-01-02

PROJE ADI: Nesneye Yönelik Yazılımlarda Kalite Ölçütlerinin İyileştirilmesi

Proje Ekibi

PROJE YÜRÜTÜCÜSÜ: Prof. Dr. Oya KALIPSIZ

ARAŞTIRMACILAR: Yrd.Doç.Dr. Yunus Emre SELÇUK
Araş.Gör. Aysun (BULUT) SEZER
Müh. Özlem ERTEMEL

Ağustos 2010

İSTANBUL

ÖNSÖZ

Yıldız Teknik Üniversitesi, Bilimsel Araştırma Projeleri Koordinatörlüğü tarafından desteklenen bu projenin amacı, yazılım kalitesini belirleyen temel iki özellikten biri olan uyumun (cohesion) ölçülebilmesi için bir ölçüt ortaya çıkarmaktır.

İÇİNDEKİLER

ÖNSÖZ 2

İÇİNDEKİLER 3

ŞEKİL LİSTESİ.....5

TABLO LİSTESİ.....6

ÖZET 7

ABSTRACT.....7

1. GİRİŞ 8

2. GENEL BİLGİLER 10

2.1. Yazılım Kalitesinin Değerlendirilmesi 10
2.2. Yazılım Kalitesinin ve Hata Eğilimliliğinin Tahmini.....11
2.3. Yazılım Modülleştirilmesi 11
2.4. Tekrar Kullanılabilir Olan Bileşenlerin Belirlenmesi 12

3. NY SİSTEMLERDE UYUMU ÖLÇMEK İÇİN KULLANILAN YAKLAŞIMLAR 12

3.1. Yapısal Uyum Ölçütleri 12
3.2. Semantik Uyum Ölçütleri 12
3.3. Entropi Tabanlı Uyum Ölçütleri 14
3.4. Slice-based Ölçütler 14
3.5. Veri Madenciliğine Dayalı Ölçütler.....15
3.6. Bilgi Tabanlı Sistemler ve Dağıtık Sistemler gibi Özelleştirilmiş Yazılım Uygulamaları için Ölçütler.....15

4. NY PROGRAMLAMADA YAPISAL BENZERLİK/UYUM ÖLÇÜTLERİ.....16

4.1. Chidamber ve Kemerer tarafından Önerilen Ölçütler.....20
4.2.4. LCOM1 20
4.2.5. LCOM2 22
4.2. Hitz ve Montazeri tarafından Önerilen Ölçütler 26
4.2.1. LCOM3 26
4.2.2. LCOM4 31
4.3. Co 37
4.4. Bieman ve Kang tarafından Önerilen Ölçütler.....39
4.5. TCC.....40
4.6. LCC.....45
4.7. Henderson-Sellors tarafından Önerilen Ölçütler.....46
4.8.4. LCOM5 46
4.8. Lee et al. tarafından Önerilen Ölçütler.....51
5.6.4. ICH.....51

5. YENİ ÖLÇÜT COMIAS 52

5.1. Metot Çağrımlarının İncelenmesi 53
5.2. Üye Alan Kullanımlarının İncelenmesi 54
5.3. Metot Çağrımı ve Üye Alan Kullanımı.....55
5.4. COMIAS Uyum Ölçütünün Tanımlanması 55
5.5. COMIAS Uyum Ölçütünün Örneklenmesi.....58
5.6. Önerilen Ölçüt ve LCOM Ölçütlerinin Bir Örnek Üzerinde İncelenmesi 65
5.6.1. LCOM1 65
5.6.2. LCOM2 65
5.6.3. LCOM3 66
5.6.4. LCOM4 67
5.6.5. LCOM5 67
5.6.6. COMIAS 68
5.7. Deneysel Geçerleme.....69
5.7.1. Kullanılan Program 69
5.7.2. Kalitesi Ölçülecek Yazılımın Seçimi 70
5.7.3. Ölçüm Sonuçları.....70
5.7.4. Zemberek Programı Ölçüm Sonuçları 72
5.7.5. Paket Seviyesinde Ölçüm Sonuçları 72
6. SONUÇLAR 76

ŞEKİL LİSTESİ

ŞEKİL 3.1: SLİCE-BASED ÖLÇÜTLER 15
ŞEKİL 4.2: POİNT SINIFININ METOT – ÜYE ALAN İLİŞKİSİ GRAFİĞİ 18
ŞEKİL 4.3: POİNT SINIFININ METOT – METOT İLİŞKİSİ GRAFİĞİ 19
ŞEKİL 4.5: SINIF A VE SINIF B METOT-ÜYE ALAN VE METOT-METOT İLİŞKİ ÖRNEĞİ 19
ŞEKİL 4.2.1: LCOM3 (1) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ GRAFİĞİ 27
ŞEKİL 4.2.2: LCOM3 (1) ŞIKKINA GÖRE GX GRAFİĞİ 28
ŞEKİL 4.2.3: LCOM3 (2) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ GRAFİĞİ 28
ŞEKİL 4.2.4: LCOM3 (2) ŞIKKINA GÖRE GX GRAFİĞİ 29
ŞEKİL 4.2.5 LCOM3 (3) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ GRAFİĞİ 29
ŞEKİL 4.2.6 LCOM3 (3) ŞIKKINA GÖRE GX GRAFİĞİ 30
ŞEKİL 4.2.7 LCOM3 SINIF A VE SINIF B'YE AİT GX GRAFİĞİ 30
ŞEKİL 4.2.8 LCOM4 (1) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ GRAFİĞİ 32
ŞEKİL 4.2.9 LCOM4 (1) ŞIKKINA GÖRE METOT ÇAĞRIMI İLİŞKİSİ GRAFİĞİ 32
ŞEKİL 4.2.10 LCOM4 (1) ŞIKKINA GÖRE GX GRAFİĞİ 33
ŞEKİL 4.2.11 LCOM4 (2) ŞIKKINA GÖRE METOT ÜYE ALAN İLİŞKİSİ 33
ŞEKİL 4.2.12 LCOM4 (2) ŞIKKINA GÖRE METOT ÇAĞRIMI İLİŞKİ GRAFİĞİ 34
ŞEKİL 4.2.13 LCOM4 (2) ŞIKKINA GÖRE GX GRAFİĞİ 34
ŞEKİL 4.2.14 LCOM4 (3) ŞIKKINA GÖRE METOT ÇAĞRIMI İLİŞKİ GRAFİĞİ 35
ŞEKİL 4.2.15 LCOM4 (3) ŞIKKINA GÖRE METOT ÇAĞRIMI İLİŞKİ GRAFİĞİ 35
ŞEKİL 4.21 LCOM4 SINIF A VE SINIF B'YE AİT GX GRAFİĞİ 35
ŞEKİL 4.5.1 TCC (1) ŞIKKINA GÖRE METOT- ÜYE ALAN VE METOT ÇAĞRIMI İLİŞKİSİ 41
ŞEKİL 4.5.2 TCC (2) ŞIKKINA GÖRE METOT-ÜYE ALAN VE METOT ÇAĞRIMI İLİŞKİSİ 42
ŞEKİL 4.5.3 TCC (3) ŞIKKINA GÖRE METOT-ÜYE ALAN VE METOT ÇAĞRIMI İLİŞKİSİ 43
ŞEKİL 4.7.1 LCOM5 (1) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ VE μ DEĞERLERİ 47
ŞEKİL 4.7.2 LCOM5 (2) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ VE μ DEĞERLERİ 48
ŞEKİL 4.7.3 LCOM5 (3) ŞIKKINA GÖRE METOT-ÜYE ALAN İLİŞKİSİ VE μ DEĞERLERİ 49
ŞEKİL 4.7.4 LCOM5 SINIF A VE SINIF B METOT-ÜYE ALAN İLİŞKİSİ VE μ DEĞERLERİ 49
ŞEKİL 5.1.2 ÖRNEK BİR DOLAYLI BAĞLANTI GRAFİĞİ G_{X2} 54
ŞEKİL 5.1.3 ÖRNEK BİR METOT ÇAĞRIMI GRAFİĞİ G_{XA} 54
ŞEKİL 5.5 MAXİMUM İLİŞKİ GRAFİĞİ ÖRNEKLERİ 59

TABLO LİSTESİ

TABLO 5.4.1 METOTLAR ARASINDA MAKSİMUM ÇAĞRIM İLİŞKİSİ ÖRNEĞİ 57
TABLO 5.4.2 METOTLAR ARASINDA ÜYE ALAN PAYLAŞIMI İLİŞKİSİ ÖRNEĞİ 57
TABLO 5.5.1 MAXİMUM İLİŞKİ GRAFİĞİ A İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 59
TABLO 5.5.2 MAXİMUM İLİŞKİ GRAFİĞİ B İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 60
TABLO 5.5.3 MİNİMUM İLİŞKİ GRAFİĞİ C İÇİN ÖLÇÜT METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 60
TABLO 5.5.4 MİNİMUM İLİŞKİ GRAFİĞİ D İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 61
TABLO 5.5.5. DOLAYLI BAĞLANTI HESABA KATILMADAN E GRAFİĞİ İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 61
TABLO 5.5.6 DOLAYLI BAĞLANTI HESABA KATILMADAN F GRAFİĞİ İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 62
TABLO 5.5.7 DOLAYLI BAĞLANTI HESABA KATILIRSA F GRAFİĞİ İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 63
TABLO 5.5.8 G GRAFİĞİ İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 64
TABLO 5.5.9 H GRAFİĞİ İÇİN METOT ÇAĞRIM VE ÜYE ALAN PAYLAŞIM DURUMU 64
TABLO 5.7.3 ZEMBEREK PROGRAMININ FARKLI SÜRÜMLERİ İÇİN GERÇEK VERİLER 71
TABLO 5.7.4 ZEMBEREK PROGRAMININ İLERLEYEN SÜRÜMLERİ İÇİN COMIAS ÖLÇÜTÜ İLE ELDE EDİLEN SONUÇLAR 72
TABLO 5.7.5 ARAÇLAR PAKETİ İÇİN COMIAS ÖLÇÜM SONUÇLARI 72
TABLO 5.7.6 YAPI PAKETİ İÇİN COMIAS ÖLÇÜM SONUÇLARI 73
TABLO 5.7.7 İŞLEMLER PAKETİ İÇİN COMIAS ÖLÇÜM SONUÇLARI 74
TABLO 6.18 İSTATİSTİK PAKETİ İÇİN COMIAS ÖLÇÜM SONUÇLARI 75

ÖZET

Yazılım ve donanım olmak üzere iki temel bileşenden oluşan bilgisayar sistemleri, günlük hayatın tüm alanlarına gittikçe artan düzeylerde dahil olmaktadırlar. Bu alanlardan bazıları sadece mali riskler taşımakta, bazıları ise hayati riskler içermektedir. Yapılan tüm çalışmalara rağmen yazılım projelerindeki başarı oranı hala yeterli düzeyde değildir. Bu nedenlerle, kaliteli yazılım üretebilmek için, yazılım geliştiricilerine doğru yönde rehberlik edecek kalite ölçütlerine gereksinim duyulmaktadır. Nesneye yönelik yazılım geliştirme ile ilgili literatür incelendiğinde, yazılım kalitesini belirleyen en önemli iki ölçütün bağlaşım (coupling) ve uyum (cohesion) olduğu anlaşılır. Bağlaşım ve uyumun ölçülmesi için önerilen ölçüt modelleri incelendiğinde, bağlaşım ölçümü açısından belirli bir olgunluğa erişilmesine rağmen, uyum ölçümünün halen yeterli düzeyde olmadığı anlaşılmıştır. Bu nedenle önerilen proje çerçevesinde yapılacak çalışmalar; bağlaşım ve uyumun ölçülmesi için bir ölçüt geliştirmek, bu ölçüt üzerinden ölçüm yapacak bir yazılım gerçeklemek ve önerilen ölçütün doğruluğunu sınamak olacaktır.

ABSTRACT

Computerized systems which consist of hardware and software components are penetrating into the many aspects of our daily lives in an increasing rate. While some of these aspects contain only financial risks, others may contain fatal and disastrous risks. Despite all efforts, the success rate of software projects is not high enough. Therefore, software developers need quality metrics that will guide them in the right direction for achieving high quality. When the object oriented programming literature is examined, it can be found that coupling and cohesion are the most significant attributes which affect software quality. After examining the cohesion and coupling metrics suggested in the literature, it can be seen that the correctness and adequateness of cohesion metrics are still open to debate while coupling metrics have reached a certain degree of maturity. Therefore the efforts to be made in the context of our projects will be to develop a metric for measuring coupling and cohesion, to implement software which will make measurements by using the developed metric and to test whether the developed metric is valid or not.

Anahtar Kelimeler: Yazılım Kalitesi, Yazılım Metrikleri, COMIAS, Coupling, Cohesion

1. GİRİŞ

Günümüzde yazılım sistemleri birçok iş alanının temel bileşenleri içerisinde yer almaktadır. Artan rekabet, gelişen teknoloji ve yazılım kuruluşlarının artan kabiliyetlerinin de etkisiyle gelişmiş yazılım sistemlerine olan ihtiyaç her geçen gün artmaktadır. Yapılan tüm çalışmalara rağmen yazılım projelerinde gerçekleşen maddi ve kimi zaman hayati başarısızlıklar engellenememiştir. Bu başarısızlığı önleyebilmek amacıyla "yazılım kalitesi" kavramı ortaya koyulmuştur. Yazılım kalitesi halen kesin sınırları belirlenememiş ve farklı kişilere farklı anlamlar ifade eden bir kavramdır. Bu da kalitenin direk değerlendirmesini zorlaştırmaktadır. Geliştiricinin bakkış açısıyla kalite, maliyet ve gecikmenin doğru tahminine, daha kolay teste, daha iyi bakıma giden yoldur. Kullanıcı tarafından bakıldığında kullanım kolaylığı ve anlaşılabilirlik gibi kavramları ifade eder (Baldassari, 2004). Yazılımın kalite özellikleri iç ve dış özellikler olarak iki gruba ayrılabilir. Dış özellikler yazılımı kullananları, iç özellikler ise yazılımı gerçekleyenleri doğrudan ilgilendirir (Peckham, 2003).

Nesneye yönelik programlarda iç kalite özelliklerinin yüksekliğini belirleyen en önemli iki özellik bağlaşım (coupling) ve uyumdur (cohesion). Bağlaşım (coupling) bir tasarımdaki iç bağımlılık derecesini, uyum (cohesion) ise tasarımın içindeki iç uyumluluğu gösterir. Nesneye Yönelik Programlama yaklaşımı için önemli bir kalite faktörü olan uyum (cohesion), bir sınıfın elemanlarının birbirine aitliğinin derecesi olarak tanımlanabilir. Bir sınıftaki metotların ve sınıfın sorumluluklarının uyumlu olması tercih edilir. Uyum eksikliği, sınıfların iki veya daha fazla alt sınıfa ayrılmaları gerekliliğini gösterir. Düşük uyum, karmaşıklığı arttırarak geliştirme sürecindeki hataların olma olasılığını arttırır. İyi yazılım tasarımı için bağlaşımı (coupling) azaltmak, uyumu ise (cohesion) artırmak gerekir.

"Nesneye Yönelik Yazılımlarda Kalite Ölçütlerinin İyileştirilmesi" isimli proje çalışmasındaki amacımız nesneye yönelik yazılımların yaşam döngüsünün her aşamasında kullanılabilecek bir uyum ölçütü ortaya koymak ve bunu deneysel olarak geçerliğini sağlamaktır. Proje kapsamında öncelikle literatür taraması derinleştirilerek önerilmiş uyum ve bağlaşım ölçütleri incelenmiş, ardından da ölçütlerin eksik yönleri belirlenmiştir. Bu bilgiler ışığında eksik bulunan yönleri iyileştirmeye yönelik çalışmalar yapılmış ve yeni bir uyum ölçütü önerilmiştir.

Yapılan literatür çalışmaları incelendiğinde çalışmaların çoğunun temelde benzer ölçütler önerdikleri görülmektedir. Birçok çalışma da daha önce gerçekleştirilenlerin tamamlayıcısı

durumundadır. Önerilen ölçütlerin birçoğunun gerçeklendirilmesi yapılmadığından nesnel sonuçlar oluşturulamamıştır. Önerilen ölçüt kümeleri arasında sadece birkaç tanesi geçerli kalmıştır. Genel olarak kabul gören ölçüt kümeleri; C&K ölçütleri (Chidamber ve Kemerer, 1994), MOOD (Brito ve Melo, 1996) ve QMOOD (Bansiya ve Davis, 2002) ölçütleridir.

C&K ölçüt kümesi çok kabul görerek, sistem tasarımcıları ve geliştiricileri için yararlı kabul edilmektedir. C&K ölçütleri sistemin sınıf seviyesinde değerlendirilebilmesini sağlar. C&K ölçütlerinin mantıksal kararlılık yönünden incelendiği bir çalışma sonucunda C&K ölçütlerinin sınıfların mantıksal kararlılığının iyi birer göstergeleri oldukları sonucuna da varılmıştır(M. Elish, D.Rine, 2003). Bununla birlikte, C&K ölçüt kümesinin bir elemanı olan ve istenmeyen bir durum olan uyum düşüklüğünün göstergesi olan LCOM ölçütünün her durumda doğru sonuç üretmediği ispatlanmıştır (Churcher ve Shepperd, 1995).

MOOD ölçütleri sistem düzeyindeki ölçütlerdir (Tobias Mayer, 1999). Herbir MOOD ölçütü, sistemdeki bir nesneye yönelik özelliğin kullanımının ölçüldüğü ve sıfır ve bir arasında değer alabilen yüzdeliklerle ifade edilebilen bir ölçüttür. C&K ölçütleri ile karşılaştırıldığında, MOOD ölçütleri sistem seviyesinde çalışan ölçütlerdir ve bu anlamda iki ölçüt kümesinin tamamlayıcı ölçütler oldukları görülmektedir. Bansiya ve Davis ise, ayrı bir ölçüttakımı olan QMOOD modelini önermiştir. Bu modelde doğrudan ölçülebilen tasarım özellikleri tanımlanmış ve bu özellikler için bir ölçüt kümesi önerilmiştir (Bansiya J., 2002). QMOOD ölçütleri üzerinde çeşitli deneysel saha çalışmaları da bulunmaktadır (Patricia L.)

Mevcut ölçüt modelleri incelendiğinde, bağlaşım ölçümü açısından belirli bir olgunluğa erişilmesine rağmen, uyumun ölçümünün halen yeterli düzeyde olmadığı anlaşılmıştır. Bu bilgilerin ışığında projenin amacı; bağlaşım ve uyumun ölçülmesi için bir ölçüt geliştirmek, bu ölçüt üzerinden ölçüm yapacak bir yazılım gerçeklemek ve önerilen ölçütün doğruluğunu sınamak olarak belirlenmiştir. Bu nedenle COMIAS (Cohesion Method Invocation Attribute Sharing – Uyum Metot Çağrımı Üye Alan Paylaşımı) adı verilen bir uyum ölçütü gerçeklenmiştir. COMIAS yapısal yaklaşım kullanılarak gerçeklenmiş bir uyum ölçütüdür. Yapısal uyum ölçütleri temel olarak sınıfın üye alanlarının üye metotlar tarafından kullanımını göz önüne almaktadır. Sınıfın metotları arasındaki ilişkiler bu düşünce ile tanımlanır. COMIAS uyum ölçütü gerçeklenirken aşağıdaki hususlar dikkate alınmıştır:

Bir sınıf üye alanlar ve üye metotlardan oluşur. Bu iki önemli elemandan birisinin ölçüt tanımında göz ardı edilmesinin, gerçek uyumun yakalanamamasına neden olacağı düşünülmektedir. Bir ölçütün normalize edilmiş olması, bir başka deyişle 0 ile 1 arasında değişen bir değer üretmesi tercih edilir. Normalize edilmiş ölçütlerin yorumlanması daha kolay olacaktır. Buna rağmen

literatürde normalize olmayan birçok ölçüt bulunmaktadır. Çoğu ölçüt tanımında dolaylı metod ilişkileri göz önüne alınmamıştır. Dolaylı bağlantılar göz ardı edildiğinde gerçek uyumun yakalanamayacağı düşünülmektedir. İyi tanımlanmış bir ölçüt mutlaka sezgilerle paralel sonuçlar vermeli, matematiksel olarak belirsiz ve tanımsız sonuçlar vermemelidir. Bir sınıfın uyumu COMIAS ölçütü kullanılarak hesaplanırken üç temel adım yürütülmektedir. Birinci adımda üye metotların arasındaki ilişki metot çağrımı yönünden incelenmekte ikinci adımda ise üye metotlar, üye alanların kullanımı yönünden incelenmekte üçüncü adımda ise ilk iki adımda elde edilen veriler birleştirilmektedir.

COMIAS uyum ölçütünün deneysel olarak geçerlenmesi amacıyla Zemberek adı verilen açık kaynak kodlu projenin çeşitli sürümleri kullanılmıştır. Zemberek, açık kaynak kodlu Türkçe doğal dil işleme kütüphanesi ve OpenOffice eklentisidir. Zemberek'in Java'da geliştirilmiş olması, kalite açısından aralarında karşılaştırma yapılabilecek farklı sürümlerinin bulunması ve nesne tabanlı olması, bu programın seçilmesinde etkili olmuştur. COMIAS tarafından üretilen sonuçların gerçek dünya verileriyle karşılaştırılabilmesi için kullanıcı görüşlerine başvurulmuştur.

2. GENEL BİLGİLER

Bu bölümde yazılım uyum ölçütlerinin hangi amaçlarla kullanılabileceği incelenmiş ve her amaç ile ilgili çeşitli değerlendirmelere yer verilmiştir.

2.1. Yazılım Kalitesinin Değerlendirilmesi

Yazılım kalitesinin değerlendirilmesi için yapılan bir çalışmada (Bansiya ve Davis, 2002), Nesneye Yönelik (NY) tasarımlarda yüksek-seviye tasarım kalitesi özelliklerini değerlendirmek için iyileştirilmiş bir hiyerarşik model tanımlanıyor. Bu modelde sınıfların, nesnelerin ve ilişkilerinin yapısal ve davranışsal tasarım özellikleri NY tasarım ölçütleri kullanılarak değerlendiriliyor.

Diğer bir çalışmada (Briand vd., 2000) amaç literatürdeki NY tasarım ölçütlerinin özellikle hata eğilimliği konusunda deneysel geçerliliğini gerçekleştirmek olmuştur. Elde edilen bir sonuç; literatürdeki çoğu ölçütün karşılaştırılabilir fikirlere ve hipotezlere dayalı olduğu ve bu nedenle gereksiz olabilecekleri görüşüdür. Uyumun hata eğilimliği üzerinde anlamlı bir etkiye sahip gözükmüyor olarak değerlendirilmiştir.

2.2. Yazılım Kalitesinin ve Hata Eğilimliliğinin Tahmini

Yazılım kalitesinin ve hata eğilimliğinin tahmini için yapılan bir çalışmada (El-Emam, 2002), NY ölçütler hata eğilimliliği anlamında inceleniyor. Elde edilen sonuçlardan bazıları şunlardır:

- Çoğu ölçüt erken tasarım safhasında toplanabiliyor ve erken kalite yönetimi sağlıyor. Yüksek değere sahip bağlaşım (coupling) ölçütlerine sahip sınıflara en iyi insanları atayabilirsiniz gibi öneriler verebilmektedir.
- Eğer tarih verisi mevcutsa, sınıfları hata eğilimliliklerine ve tahmin edilen sonuçlarına göre derecelendirmek iyi olacaktır. Bunun sonucu olarak da hata eğilimliliği en yüksek tahmin edilen sınıflara en iyi insanları atayabilirsiniz gibi önerilerde bulunulabilir.
- Yüksek hata eğilimliğine sahip sınıflara geniş ve deneyimli inceleme ekipleri atamalı ve bu sınıflar için daha fazla test adımı geliştirmelidir.
- Var olan uyum ölçütleri hata eğilimliliği tahmininde iyi değildirlerdir.

Yazılım kalitesinin ve hata eğilimliğinin tahmini için yapılan diğer bir çalışmada (Quah ve Thwin, 2003), NY ölçütleri kullanarak yazılım kalitesi tahmininde neural network'ün uygulanması incelenmiştir. Kalite tahmini yazılım bakım yapılabilirliğinin yanı sıra güvenilirliğinin tahminini de içerir. Güvenilirlik hata sayısı olarak ölçülebilir. Bakım eforu ise sınıf başına değişen satır sayısı olarak ölçülebilir. İlki için bir sınıftaki hata sayısını ikincisi için sınıf başına düşen değişiklik sayısını tahmin etmek gerekiyor. Bu çalışmada, 2 neural network modeli kullanılıyor, Ward ve General Regression (GRRN). Miras ile ilişkili ölçütler, karmaşıklık ölçütleri, uyum ölçütleri, bağlaşım ölçütleri ve bellek yerleştirme ölçütleri bağımsız değişken olarak kullanılıyor. GRRN network modelinin Ward 'dan daha doğru tahminler yapabildiği görülmüştür.

2.3. Yazılım Modülleştirilmesi

Yazılım modülleştirilmesi için yapılan bir çalışmada (Brito e Abreu ve Goulao, 2001), daha iyi modülleştirme çözümlerini elde etmek için kümeleme analizi tekniklerini ve sınıf bağlaşımı (class-coupling) bilgisini kullanan bir yaklaşım öneriliyor. Bu yaklaşım yapısal bağlaşımı göz önünde bulunduruyor fakat orijinal sistem modülleştirmesinin arkasındaki semantiğin bir kısmını da kullanıyor. En iyi çözümü bulmak için birkaç kümeleme tekniği ve bağlaşım bilgisinin birkaç kombinasyonu denenmiş. Buradaki yaklaşım hem tasarım safhasında hem de reengineeringde yanlış modülleştirmeleri ve alternatif olanları belirlemek için kullanılabilir. Bu çalışmada geliştirilen araç, kalite anlamında yararlı bir destek sağlıyor.

Yazılım modülleştirilmesi için yapılan diğer bir çalışmada (Maletic ve Marcus, 2001), programların semantiğinin ve yapısal bilgisinin birlikte kullanımının, bakım ve reengineering'deki işleri desteklemesi sorgulanmaktadır. Burada semantik, domain (problem ve geliştirme) özgü meseleleri, yapısal ise programın gerçek sintatik yapısını adresliyor. Önemli bir bilgi alma metodu olan "Latent semantic indexing", yazılım bileşenleri arasındaki semantik benzerliği tanımlamada kullanılıyor. Daha sonra Yazılım sistemi içindeki bileşenler benzerlik ölçülerine göre kümeleniyor. Basit yapısal bilgi (ör. dosya organizasyonu) daha sonra kümelerin ve dosyaların semantik uyumu açısından değerlendirilmesinde kullanılıyor. Yapılan deneyler sonucunda; kaynak kod dokümanlarının semantik benzerliğinin yazılım bakımı ve evrimi konularında değerli bilgiler sağladığı görülmüştür.

2.4. Tekrar Kullanılabilir Olan Bileşenlerin Belirlenmesi

Tekrar kullanılabilir olan bileşenlerin belirlenmesi ile ilgili yapılan bir çalışmada (Etzkorn ve Davis, 1997), şu konulara değiniliyor; NY kod miras yoluyla fonksiyonel koddan daha fazla tekrar kullanılabilir yapıda olmasına rağmen çoğu NY kod tekrar kullanım mantığı ile yazılmıyor, bu da yararlı bileşenleri belirlemeyi zorlaştırıyor. Diğer bir çalışmada ise (Lee vd., 2001), Patricia sistemi adıyla bir sistem geliştirilmiştir. Bu sistem, sezgisel bir yaklaşım kullanarak otomatik olarak yorumları ve belirteçleri anlayarak bu bileşenleri belirlemektedir.

3. NY SİSTEMLERDE UYUMU ÖLÇMEK İÇİN KULLANILAN YAKLAŞIMLAR

NY sistemlerde uyumu ölçmek için kullanılan farklı yaklaşımlar vardır. Bu yaklaşımlar altta yatan mekanizmaya göre şu şekilde ayrıştırılabilir (Poshyvanyk ve Marcus, 2005):

3.1. Yapısal Uyum Ölçütleri

Yapısal ölçütlerin arkasındaki düşünce, metotlar arasındaki sınıf değişken referanslaması ve paylaşımını göz önüne almaktır. Çoğu yapısal ölçütler, sınıfın metotları arasındaki ilişkileri bu düşünce ile tanımlar ve ölçer. Uyumun, üye alanlarını veya sınıf değişkenlerini paylaşan metodt çiftlerinin sayısına dayandığı görülüyor. Yapısal ölçütlerin arasındaki farklar, metotlar arası ilişkilerin tanımına, sistem tanımına ve sayma mekanizmasına dayanıyor.

3.2. Semantik Uyum Ölçütleri

Yapılan bir çalışmada (Etzkorn ve Delugach, 2000) sınıf benzerliğini, sınıf tanım kümesi karmaşıklığını, anahtar sınıf belirlemesini, sınıf arayüz karmaşıklığını, sınıf üst üste binmesini ve sınıfın dokümantasyon kalitesini ölçen mantıksal tasarım ölçütleri önerilmiştir.

3.2.1. Sınıf benzerliği (Class cohesion):

- Logical Relatedness of Methods (LORM): sınıfın metotlarının kavramsal ilişkiliğini ölçer. Bu ölçümü toplamak için uzman sistem tarafından semantik analiz uygulanmalıdır.
- LORM, sınıftaki toplam ilişki sayısı / sınıftaki mümkün olan toplam ilişki sayısı şeklinde hesaplanır.
- LORM2, birden fazla fonksiyonda görünen kavramları inceleyerek fonksiyonelliğin üst üste binmesini ölçer.
- LORM3, kavram benzerliğini ölçer.

3.2.2. Sınıf tanım kümesi karmaşıklığı (Class domain complexity):

- Class Domain Complexity (CDC)
- Semantic Class Definition Entropy (SCDE)

3.2.3. Anahtar sınıf belirleme (Key Class Identification):

- KCI: 0 veya 1 değerini alır (RCDC<0.75 ise 0, değilse 1; RCDC maximum tanım kümesi karmaşıklığı içeren sınıf 1 RCDC'ye sahiptir.)

3.2.4. Sınıf arayüz karmaşıklığı (Class Interface Complexity):

- CIC: CDC'ye benzer fakat sınıfın arayüz metodları üzerinde çalışır.
- SCIDE: arayüz karmaşıklığını ölçer.

3.2.5. Sınıf üst üste binmesi (Class overlap):

Birden fazla sınıfta görünen kavramları inceleyerek 2 sınıf arasındaki fonksiyonelliğin üst üste binmesini ölçerler.

- Class Overlap A (COa)
- Class Overlap B (COb)

3.2.6. Sınıfın dokümantasyon kalitesi (Documentation):

Sınıfın belirteçlerini ve yorumlarını kullanarak ölçüm yaparlar. Bu ölçütlerin bir önceki bölümlerde anlatılan ve geneli sintatik olan ölçütlerden daha doğru sonuçlar verebildikleri görülmüştür (Etzkorn ve Delugach, 2000).

- Overall Class Documentation Quality A (OCDQa)
- Overall Class Documentation Quality B (OCDQb)
- Class Comment Quality (CCQ)
- Class Identifier Quality (CIQ)

3.3. Entropi Tabanlı Uyum Ölçütleri

Yazılım mimaricileri ve tasarımcıları yazılımın soyut gösterimi için birçok grafik çeşidi kullanırlar. Örneğin "çağrı grafikleri (call graphs)" birçok yazılım geliştirici tarafından potansiyel "çağrı ilişkileri (calling relationships)"ni göstermek için kullanılır. Bilgi teorisi yaklaşımı, bir grafikteki kalıpları (patterns) saymaktan daha farklı bir yaklaşımdır. Ölçüt tanımları, grafiğin elemanlarını saymaktan ziyade ilişki kalıplarını göz önüne alıyor. Her düğüme bağlı kenarların kalıbı rastgele bir sayı olarak modelleniyor. Deneysel bir çalışmada (Allen vd., 2001) bir bilgisayar oyunu incelenmiş, kaynak kodu modül olarak bilinen bir fonksiyon çağrı grafiğinin bağlaşım ve uyumu hesaplanmış. Sonuç olarak bu ölçütlerin standart sayı tabanlı ölçütlerle bezer sonuçlar verdiği görülmüştür.

3.4. Slice-based Ölçütler

Yazılımın yeniden yapılandırılması maliyetli bir iştir. Uyum ölçütleri, belirleme ve ölçme işinde yardım edecek potansiyele sahiptir. Böyle ölçütler de geniş çalışmalarda slice-based ölçütlerdir. Mark Weiser (Meyers ve Binkley, 2004) program slicing tabanlı 5 ölçüt önermiştir (Şekil 3.1). Bu ölçütlerin deneysel incelemeleri sonucu bazı katkılarının olduğu görülmüştür:

- Ölçütler, modül seviyesinde kullanılabildikleri için reverse engineering işinde direk modüle odaklanmayı sağlayarak yardımcı olur. Slice-based ölçütler için baseline değerler sağlanmıştır. Bu değerler, reengineering eforunun hedef değerin dışındaki modüller üzerinde olması gerektiğini vurgular.
- Slice-based ölçütlerin programın yaşlandıkça bozulma miktarını belirleyebildikleri görülmüştür.
- Ölçütler, program seviyesinde kullanılabilerek reverse engineering, kodu iyileştirme işinde rehberlik ederler.

Metric	Definition	Description
$Tightness(M) =$	$\frac{\mid SL_{int} \mid}{length(M)}$	Tightness measures the number of statements in every slice.
$MinCoverage(M) =$	$\frac{1}{length(M)} \min_i \mid SL_i \mid$	MinCoverage is the ratio of the smallest slice in a module to the module's length.
$Coverage(M) =$	$\frac{1}{\mid V_O \mid} \sum_{i=1}^{\mid V_O \mid} \frac{\mid SL_i \mid}{length(M)}$	Coverage compares the length of slices to the length of the entire module.
$MaxCoverage(M) =$	$\frac{1}{length(M)} \max_i \mid SL_i \mid$	MaxCoverage is the ratio of the largest slice in a module to the module's length.
$Overlap(M) =$	$\frac{1}{\mid V_O \mid} \sum_{i=1}^{\mid V_O \mid} \frac{\mid SL_{int} \mid}{\mid SL_i \mid}$	Overlap is a measure of how many statements in a slice are found only in that slice.

Şekil 3.1: Slice-based ölçütler

3.5. Veri Madenciliğine Dayalı Ölçütler

Yapılan bir çalışmada (Montes de Oca ve Carver, 1998) veri uyumlu (cohesive) alt sistemleri belirlemek için veri madenciliğini kullanan ISA metodolojisi tanımlanıyor. Bu yaklaşımı kullanarak COBOL sistemler alt sistemlere ayrıştırılabilmiştir. Bu çalışmada, veri madenciliğinin farklı tanecik seviyesinde uygulanabileceği ve framework olarak kullanılabilecek bir 3 adım metodu önerilmiş. Önerilen adımlar şunlardır;

- Sistemin veri tabanı görünüşünü tanımla.
- Veri madenciliğini gerçekleştir.
- Sonuçları birleştir ve yorumla.

Deneyimler göstermiştir ki, veri madenciliği bir yazılım sisteminin mantıksal ayrıştırılmasının üretiminde kullanılabilir. Veri madenciliği, sistemin hiç bir konu ve geçmiş bilgisini bilmeden veri uyumlu alt sistemleri belirleyebilir. Sadece gerekli olan şey kaynak koddur. Veri madenciliğinin yararlı özellikleri büyük dokümante edilmemiş sistemlerin analizinde uygun bir yaklaşım olduğunu göstermiştir.

3.6. Bilgi Tabanlı Sistemler ve Dağıtık Sistemler gibi Özelleştirilmiş Yazılım Uygulamaları için Ölçütler

Yapılan bir çalışmada (Kramer ve Kaindl, 2004) Bilgi tabanlı sistemler için uyum ölçütü, ChF (Cohesion of Frame) tanımlanmış. Bu ölçüt bilgi tabanlarının modülaritesi ile ilgili problemleri adreslemede yardımcı olmuş. Bilgi tabanlı sistemlerin geliştirici tarafından nicel ve nesnel kalite özelliklerinin sorgulanmasına ve sistemin geliştirilmesine yardımcı olabileceği düşünülmüştür.

İstemci/sunucu ve dağıtık uygulamalarda nesne kalite kümelemesi, sistemin bütününün performansına karar vermede anahtar bir rol oynar. Bu nedenle, yüksek bağlaşıma sahip nesneler tek bir kümede gruplanmalı ki her bir küme yüksek bir uyuma sahip olsun. Sonuç olarak, nesneler arası mesaj trafiği yüksek oranda azaltılabilir. Çoğu ölçütler nesne mesaj trafik yükünü ölçmek için uygun değildirler. Yapılan bir çalışmada çeşitli sınıf ilişkileri üzerindeki ilgili ağırlıkları göz önüne alan ve nesneler arasındaki statik ve dinamik mesaj akışını tahminleyen yeni bir ölçüt kümesi öneriliyor. Yapılan değerlendirmelerde bu ölçütlerin, dinamik ölçümü de yapabildiği görülmüştür.

4. NY PROGRAMLAMADA YAPISAL BENZERLİK/UYUM ÖLÇÜTLERİ

Bu bölümde literatürde önerilen yapısal uyum ölçütleri anlatılmakta, Örnek 1 ve Örnek 2 üzerinde her bir ölçütün uygulaması yapılarak değerlendirilmeler verilmektedir.

Örnek 1, Java kaynak kodlu bir program örneğidir (Şekil 4.1).

```
public class Point
{
   private Boolean Visible;
   private int x, y;

   public Point(int InitX, int InitY){
      this.x = InitX;
      this.y = InitY;
      this.Visible = false;
   }
   public int GetX(){
      return x;
   }
   public int GetY(){
      return y;
   }
   public void SetX(int x){
      this.x = x;
   }
   public void SetY(int y){
```

```
    this.y = y;
}
public void Show(){
    Visible = true;
    System.out.println("X : " + x + "  Y : " + y);
}
public void Hide(){
    Visible = false;
}
public void Drag(int DragBy){
    Show();
    int delta = 1;
    int counter = DragBy / delta;
    for (int i = 0; i < counter; ++i){
        MoveTo( x + delta, y + delta  );
    }
}
public Boolean IsVisible(){
    return Visible;
}
public void MoveTo( int NewX, int NewY){
    Hide();
    x = NewX;
    y = NewY;
    Show();
}
```

```
}
```

Şekil 4.1: Bir java program örneği 1

Şekil 2'deki kod örneğinde bulunan tüm öğeleri inceleyelim:

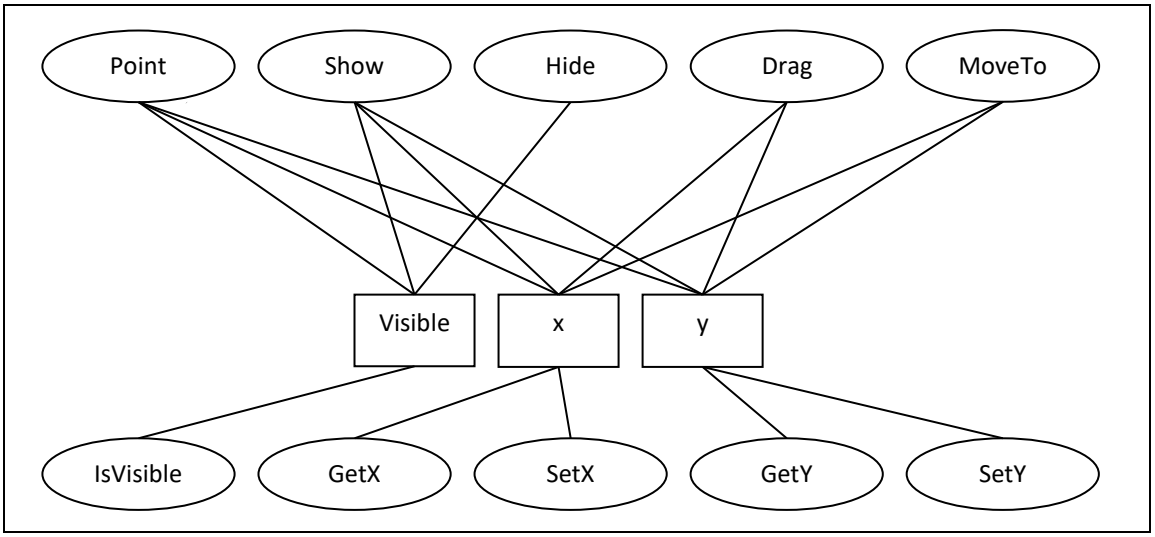
Sınıf: Point

Contructor metot: Point()

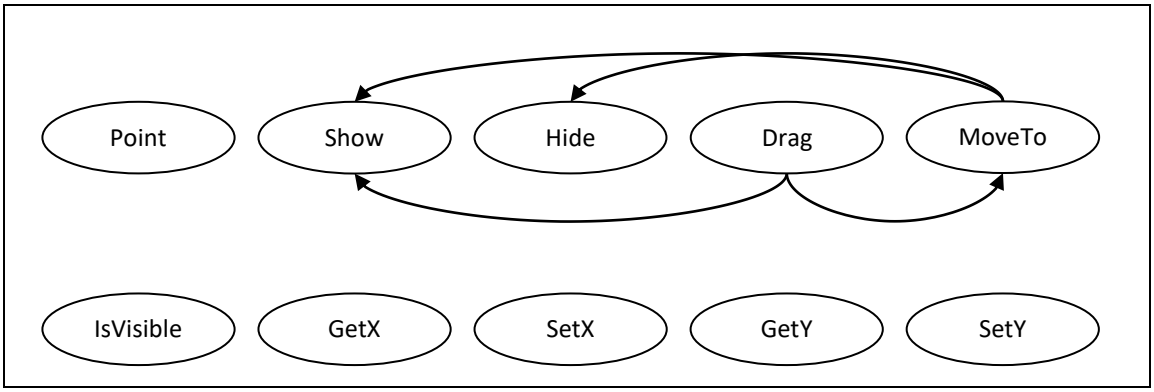
Erişim metotları: IsVisible(),GetX(), GetY(), SetX(), SetY()

Normal metotlar: Show(), Hide(), Drag(), MoveTo()

Point sınıfının metotları ile eriştikleri üye alanların grafiksel gösterimi Şekil 4.2'de, metotların birbiriyle ilişkileri ise Şekil 4.3'de verilmektedir. metot – üye alan ilişkisi gösteriminde metot ve eriştiği üye alan arasında bir çizgi oluşturulmuş, metot – metot ilişkisi gösteriminde ise çağıran metottan çağrılan metoda doğru yönlü bir çizgi oluşturulmuştur.

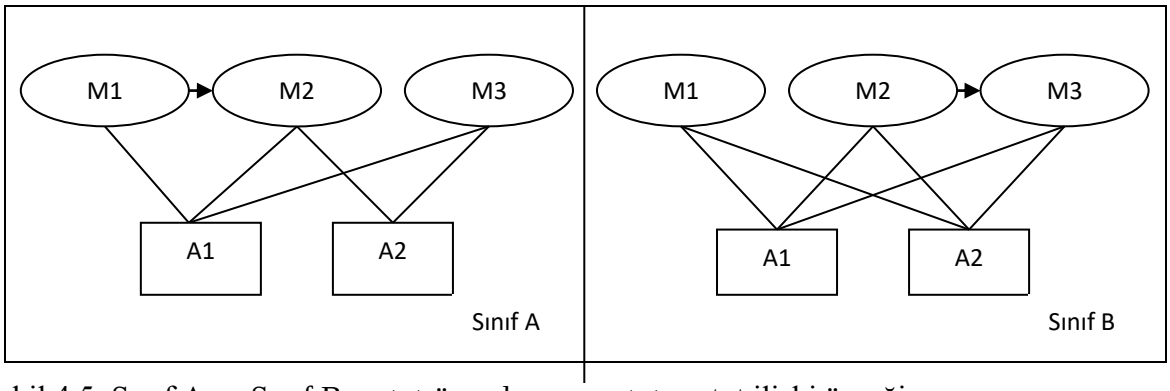


Şekil 4.2: Point sınıfının metot – üye alan ilişkisi grafiği



Şekil 4.3: Point sınıfının metot – metot ilişkisi grafiği

Örnek 2 ise iki adet sınıfın metot – üye alan ve metot – metot ilişki diyagramlarının bulunduğu bir örnektir (Şekil 4.5).



Şekil 4.5: Sınıf A ve Sınıf B metot-üye alan ve metot-metot ilişki örneği

Miras alınmış metot ve üye alanları ölçüt hesaplamasına almak tamamen isteğe bağlı bir durumdur. Bu konu ile ilgili olarak Bieman ve Kang (Zhou vd., 2002) bir sınıfın uyumunun analizinde kullanılabilecek 3 seçenek önermişlerdir:

- Miras alınmış metotları ve üye alanları hesaplamaya almamak
- Miras alınmış metotları ve üye alanları hesaplamaya almak
- Miras alınmış metotları hesaplamaya almamak ve üye alanları hesaplamaya almak

Bu 3 seçeneğe ilave olarak Briand ve arakadaşları (Briand vd., 1998) şu seçeneği önermişlerdir:

- Miras alınmış metotları hesaplamaya almak ve üye alanları hesaplamaya almamak

Ölçütlerin bazıları özellikle miras durumunu hesaba katmayı belirtirken çoğu ise hiçbir belirtimde bulunmamıştır. Bu nedenle yapılacak ölçümlerde miras göz önüne alınmamaktadır. Bir sınıftaki bir metot eğer metot erişim metodu, delegasyon metodu, constructor ya da destructor metot

ise *özel*dir (Chae ve Kwon, 1998). Bunun dışında kalan metotlar *normal* metotlardır. *Erişim metodu*, işi sadece üye alan almak veya güncellemek olan metottur. Sonuç olarak sadece bir üye alan referans eder. *Delegasyon metodu*nun işi, bir başka nesneye özellikle sınıftaki bir üye alana mesaj delege etmektir, böylece bir üye alan ile bir etkileşime sahiptir.

Erişim metotları diğer metotların direk üye alana referans etme ihtiyacını ortadan kaldırıp kendi üzerlerinden eriştirdikleri için yapay olarak uyumu düşürürler, constructorlar çoğu üye alanın ilk değerlerini sağladıkları için yapay olarak uyumu arttırırlar.

Bu bilgiler ışığında ölçütlerin Örnek 1 ve Örnek 2 uygulamaları yapılmaktadır. Metot tipinin *uyum*'a etkisini değerlendirebilmek amacıyla her bir ölçüt için 3 farklı yönden ölçüm yapılmaktadır:

- Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak
- Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak
- Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

4.1. Chidamber ve Kemerer tarafından Önerilen Ölçütler

4.2.4. LCOM1

LCOM1 (Lack of Cohesion of Methods) ölçütü, Bunge'ın "similarity" ontolojisine dayanır. Uyum bu ontolojiye göre "metotların benzerlik derecesi" şeklinde tanımlanıyor.

LCOM1, Chidamber ve Kemerer tarafından 1991 yılında tanımlanan bir ölçüttür (Chidamber ve Kemerer, 1991).

Tanım :

M1, M2, ... , Mn metoda sahip bir C sınıfı olsun. {Ii} = Mi metodu tarafından kullanılan üye alanların kümesi olsun. Bu durumda n tane {I1}, {I2} ... , {In}küme olur. LCOM1, bir sınıftaki ortak üye alanı bulunmayan metot çifti sayısı olarak ölçülüyor.

LCOM1 ölçütü ters uyum ölçümüdür. Yüksek LCOM1 değeri düşük uyumu gösterirken düşük LCOM1 değeri yüksek uyumu gösterir.

LCOM1 Ölçütü Hesaplaması:

Örnek 1:

1) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

I_4: {x, y} I_5: {x, y} I_6: {Visible}

M_7: GetX() M_8:SetX() M_9: GetY() M_{10}: SetY()

I_7: {x} I_8: {x} I_9: {y} I_{10}: {y}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9},{M_9, M_{10}},

LCOM1 = 16

2) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

I_4: {x, y} I_5: {x, y}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}

LCOM1 = 2

3) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

I_5: {x, y} I_6: {Visible} I_7: {x,}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9},{M_9, M_{10}},

LCOM1 = 16

Örnek 2:

Sınıf A için; LCOM1 = 0

Sınıf B için; LCOM1 = 0

Değerlendirme:

- LCOM1 ölçütünün minimum değeri "0" ve maximum değeri metot çifti sayısı ile değişiyor. LCOM1 normalize edilmiş bir ölçüt değildir.
- Metot çağrımları göz önüne alınmamıştır.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Sadece 1 metoda sahip sınıf için LCOM1 uygulanamıyor.
- Teorik olarak geçerlenmiştir.
- LCOM1 ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 1 için yapılan LCOM1 ölçümü sonucunda erişim metotlarının, *uyumu*, yapay olarak düşürdüğü gözlemlenirken constructor metotların göz önüne alınmasının ölçüm sonuçlarında herhangi bir değişikliğe yol açmadığı görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak LCOM1 hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize LCOM1 ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da LCOM1 ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.

4.2.5. LCOM2

LCOM2, Chidamber ve Kemerer tarafından 1994 yılında tanımlanan bir ölçüttür (Chidamber ve Kemerer, 1994). LCOM1 ölçütünden yola çıkılarak önerilmiştir.

Tanım :

M_1, M_2, ... , M_n metoda sahip bir C sınıfı olsun. $\{I_i\} = M_i$ metodu tarafından kullanılan üye alanların kümesi olsun. Bu durumda n tane $\{I_1\}$, $\{I_2\}$ ... , $\{I_n\}$küme olur. P = $\{(I_i, I_j) | I_i \cap I_j = \phi\}$ ve Q = $\{(I_i, I_j) | I_i \cap I_j \neq \phi\}$ olsun. Eğer bütün $\{I_1\}$, $\{I_2\}$ ... , $\{I_n\}$ kümeleri = ϕ ise P = ϕ olur. LCOM2 şu şekilde tanımlanıyor:

$$\text{LCOM2} = \begin{cases} |P| - |Q|, if\ |P| > |Q| \\ 0, if\ |P| \leq |Q| \end{cases} \qquad (4.1)$$

Diğer bir deyişle, LCOM2 bir sınıftaki ortak üye alanı bulunmayan metot çifti sayısından ortak üye alanı bulunan metot çifti sayısının çıkarılmasıyla ölçülüyor.

LCOM2 ölçütü ters uyum ölçümüdür. Yüksek LCOM2 değeri düşük uyumu gösterirken düşük LCOM2 değeri yüksek uyumu gösterir.

LCOM2 Ölçütü Hesaplaması:

Örnek 1:

1) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

I_4: {x, y} I_5: {x, y} I_6: {Visible}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9},{M_9, M_{10}},

P = 16

Ortak üye alanı bulunan metot çiftleri:

{M_1, M_2}, {M_1, M_3}, {M_1, M_4}, {M_1, M_5}, {M_1, M_6}, {M_1, M_7}, {M_1, M_8}, {M_1, M_9}, {M_1, M_{10}}, {M_2, M_3}, {M_2, M_4}, {M_2, M_5}, {M_2, M_6}, {M_2, M_7}, {M_2, M_8}, {M_2, M_9}, {M_2, M_{10}}, {M_3, M_6}, {M_4, M_5}, {M_4, M_7}, {M_4, M_8}, {M_4, M_9}, {M_4, M_{10}}, {M_5, M_7}, {M_5, M_8}, {M_5, M_9}, {M_5, M_{10}}, {M_7, M_8}, {M_9, M_{10}}

Q = 29

P < Q, LCOM2 = 0

2) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

I_4: {x, y} I_5: {x, y}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}

P = 2

Ortak üye alanı bulunan metot çiftleri:

{M_1, M_2}, {M_1, M_3}, {M_1, M_4}, {M_1, M_5}, {M_2, M_3}, {M_2, M_4}, {M_2, M_5}, {M_4, M_5}

Q = 8

P < Q, LCOM2 = 0

3) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

I_5: {x, y} I_6: {Visible} I_7: {x,}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}

Ortak üye alanı bulunmayan metot çiftleri:

{M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9},{M_9, M_{10}}

P = 16

Ortak üye alanı bulunan metot çiftleri:

{M_2, M_3}, {M_2, M_4}, {M_2, M_5}, {M_2, M_6}, {M_2, M_7}, {M_2, M_8}, {M_2, M_9}, {M_2, M_{10}}, {M_3, M_6}, {M_4, M_5}, {M_4, M_7}, {M_4, M_8}, {M_4, M_9}, {M_4, M_{10}}, {M_5, M_7}, {M_5, M_8}, {M_5, M_9}, {M_5, M_{10}}, {M_7, M_8}, {M_9, M_{10}}

Q = 20

P < Q, LCOM2 = 0

Örnek 2:

Sınıf A için;

Ortak üye alanı bulunmayan metot çiftleri: {-}

P = 0

Ortak üye alanı bulunan metot çiftleri:

{M_1, M_2}, {M_1, M_3}, {M_2, M_3}

P = 3

P < Q, LCOM2 = 0

Sınıf B için;

Ortak üye alanı bulunmayan metot çiftleri: {-}

P = 0

Ortak üye alanı bulunan metot çiftleri:

{M_1, M_2}, {M_1, M_3}, {M_2, M_3}

P = 3

P < Q, LCOM2 = 0

Değerlendirme:

- LCOM2 ölçütünün minimum değeri "0" ve maximum değeri metot çifti sayısı ile değişiyor. LCOM2 normalize edilmiş bir ölçüt değildir.

- Metot çağrımları göz önüne alınmamıştır.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Sadece 1 metoda sahip sınıf için LCOM2 uygulanamıyor.
- Teorik olarak ve deneysel olarak (Briand vd., 1998) geçerlenmiştir.
- LCOM2 ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 1 için yapılan LCOM2 ölçümü sonucunda, erişim metotlarının ve constructor metotların göz önüne alınmasının *uyumu* etkilemediği görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak LCOM2 hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize LCOM2 ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da LCOM2 ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.

4.2. Hitz ve Montazeri tarafından Önerilen Ölçütler

4.2.1. LCOM3

Hitz ve Montazeri tarafından 1995 yılında tanımlanan ölçüttür (Hitz ve Montazeri, 1995). LCOM1 ölçütünün değişik bir varyasyonudur.

Tanım :

X bir sınıf olsun. I_x, X sınıfının üye alanlarının kümesi olsun. M_x, X sınıfının metotlarının kümesi olsun. Yönsüz bir grafik $G_x(V, E)$ göz önüne alalım, uçları sınıfın metotları olsun.

$V = M_x$ (4.2)

$E = \left\{ < m, n > \in V \times V \middle| \exists i \in I_X : (\text{m accesses i}) \wedge (n \text{ accesses i}) \right\}$ (4.2)

LCOM3, G_x grafiğinin bağlı bileşenlerinin sayısı olarak tanımlanıyor;

$G_x(1 \leq LCOM3 \leq |M_X|)$. (4.4)

Diğer bir ifadeyle, Yönsüz bir grafik G göz önüne alalım, uçları sınıfın metotları olsun. Eğer metotlar en az bir ortak üye alan kullanıyorlarsa aralarında bir kenar olsun. Bu durumda LCOM3, G grafiğinin bağlı bileşenlerinin sayısı olarak tanımlanıyor.

LCOM3 ölçütü ters uyum ölçümüdür. Yüksek LCOM3 değeri düşük uyumu gösterirken düşük LCOM3 değeri yüksek uyumu gösterir.

LCOM3 Ölçütü Hesaplaması:

Örnek 1:

4) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

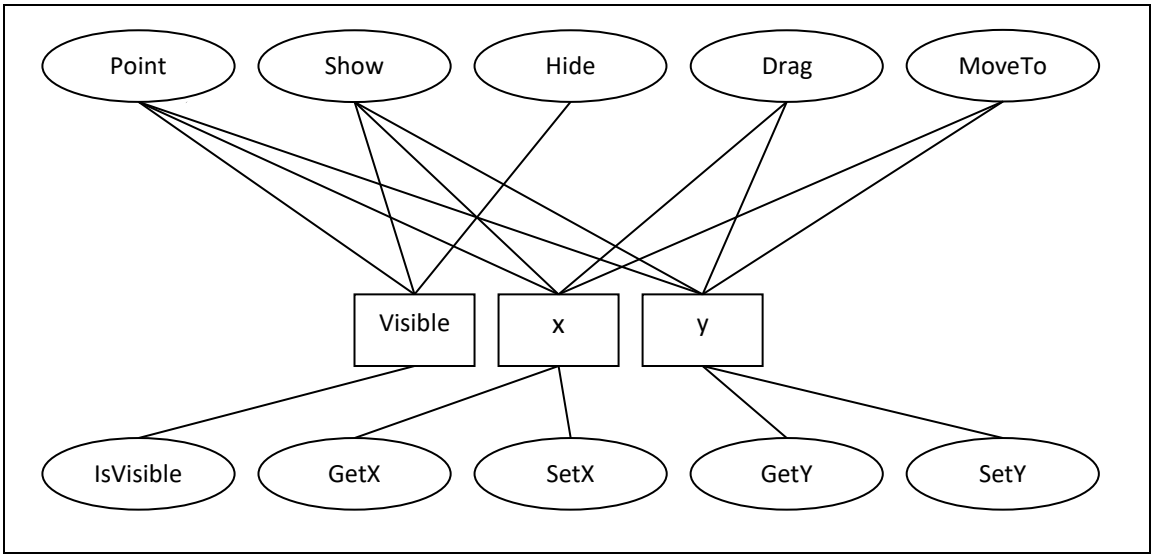
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

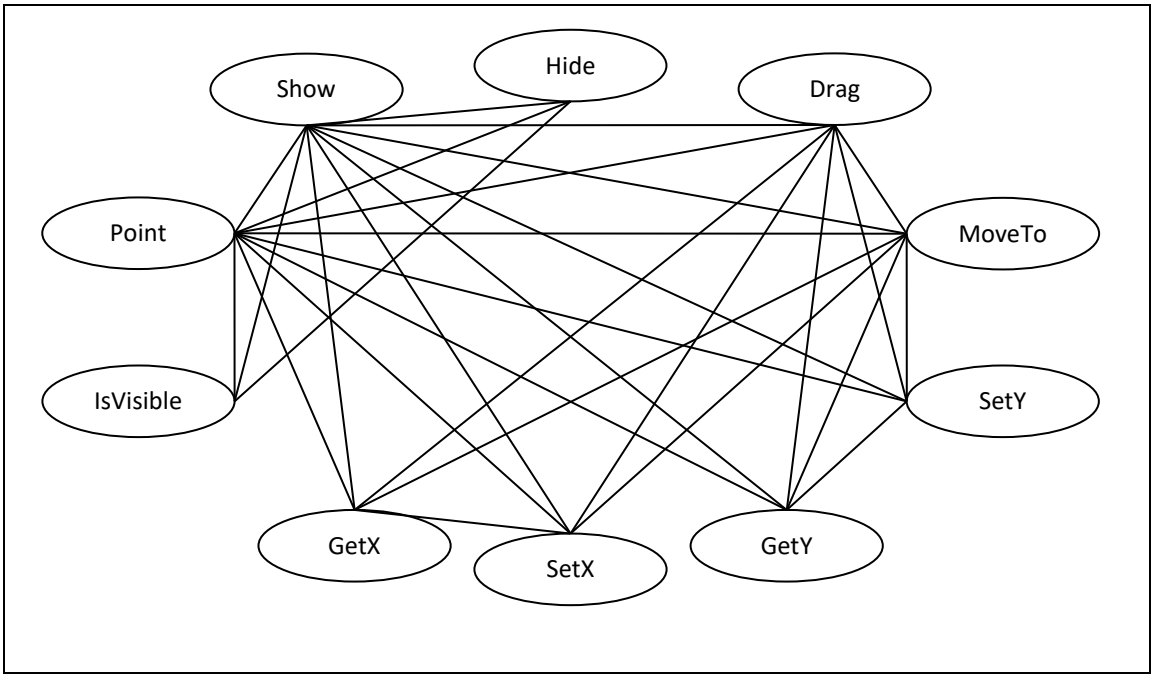
I_4: {x, y} I_5: {x, y} I_6: {Visible}

M_7: GetX() M_8:SetX() M_9: GetY() M_{10}: SetY()

I_7: {x} I_8: {x} I_9: {y} I_{10}: {y}



Şekil 4.2.1: LCOM3 (1) şıkkına göre metot-üye alan ilişkisi grafiği



Şekil 4.2.2: LCOM3 (1) şıkkına göre Gx grafiği

LCOM3 = 1

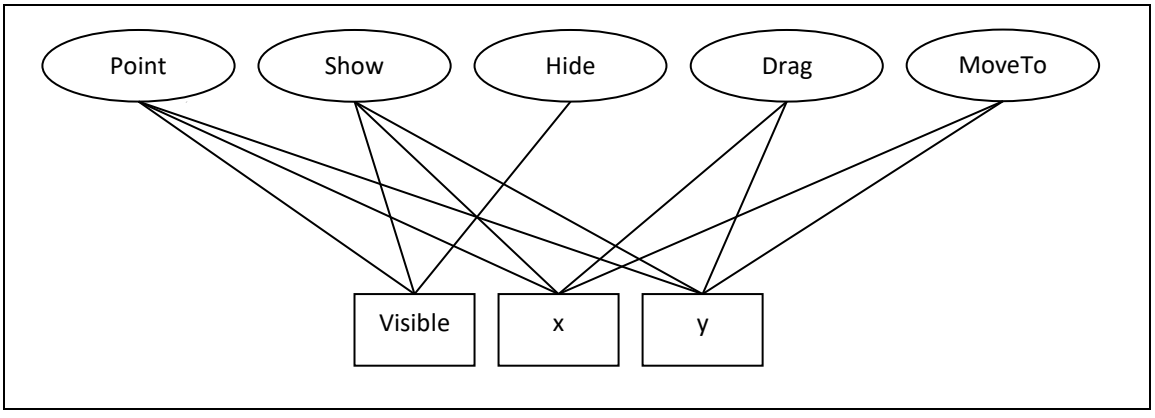
5) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

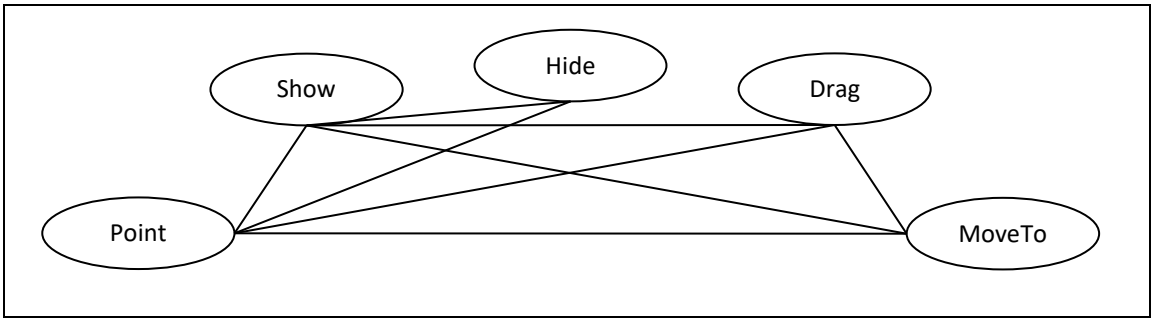
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

I_4: {x, y} I_5: {x, y}



Şekil 4.2.3: LCOM3 (2) şıkkına göre metot-üye alan ilişkisi grafiği



Şekil 4.2.4: LCOM3 (2) şıkkına göre Gx grafiği

LCOM3 = 1

6) Constructor metotları hesaplamaya alınmayacak, erişim metotlar hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

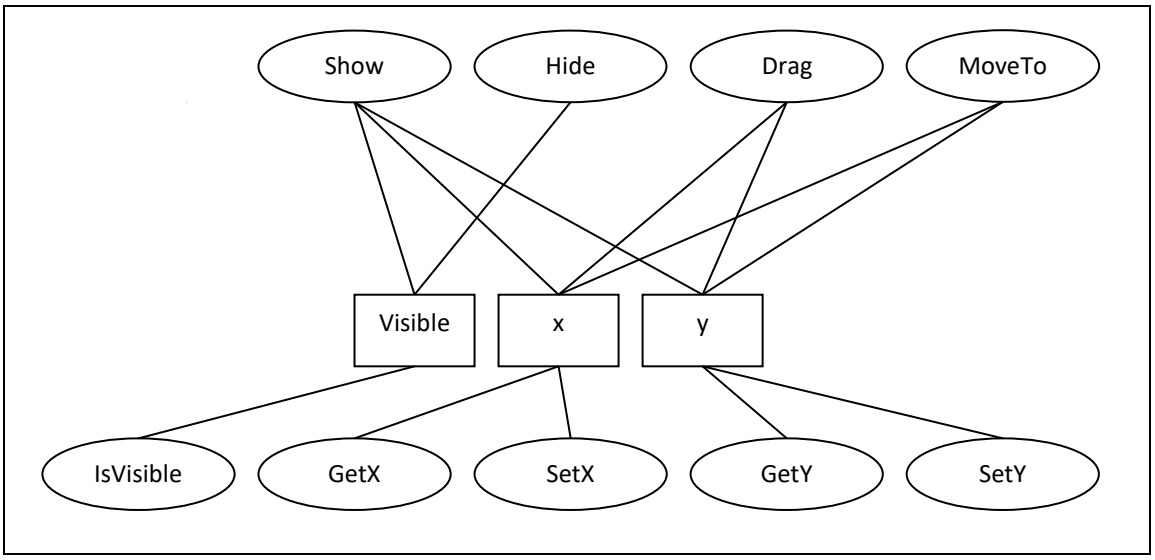
I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

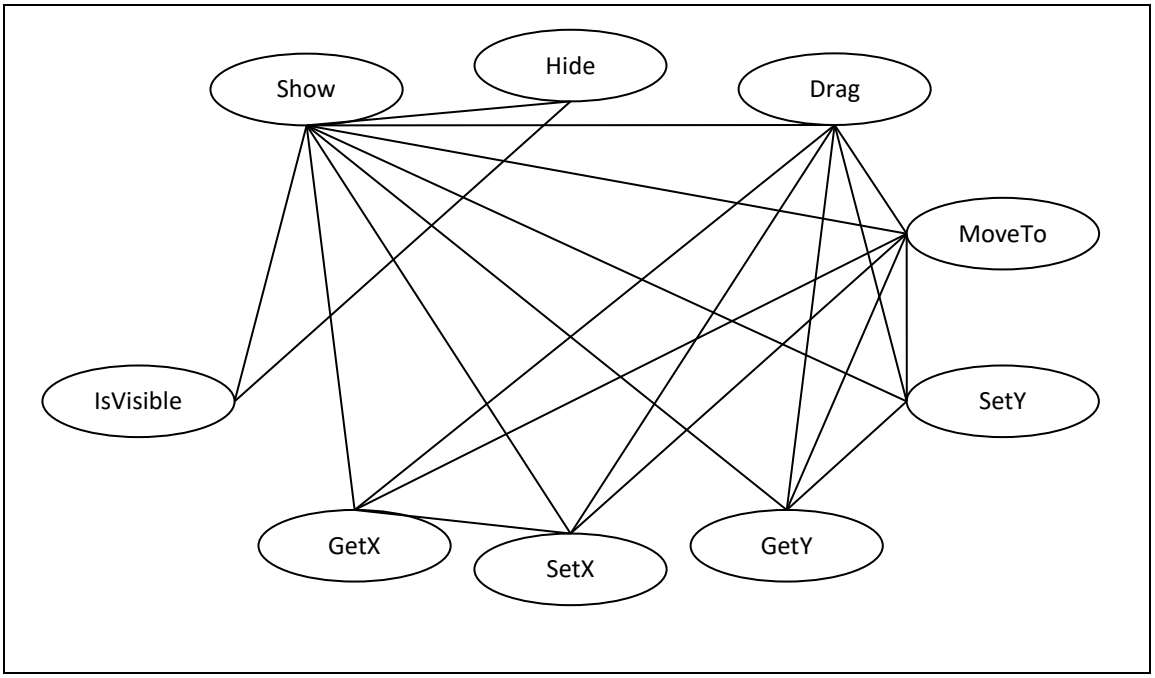
I_5: {x, y} I_6: {Visible} I_7: {x,}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}



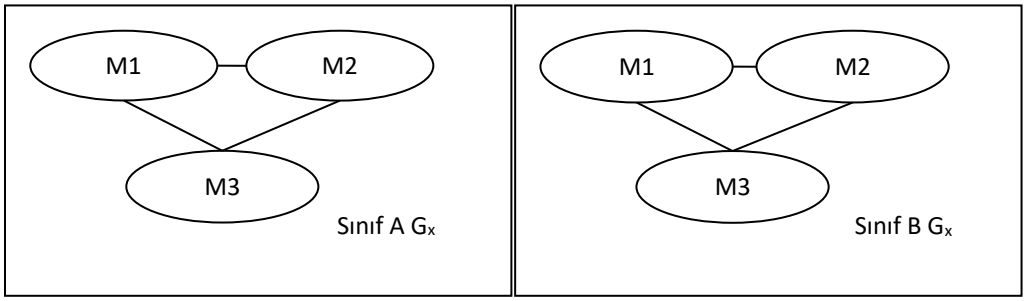
Şekil 4.2.5 LCOM3 (3) şıkkına göre metot-üye alan ilişkisi grafiği



Şekil 4.2.6 LCOM3 (3) şıkkına göre Gx grafiği

LCOM3 = 1

Örnek 2:



Şekil 4.2.7 LCOM3 Sınıf A ve Sınıf B'ye ait Gx grafiği

Sınıf A için; LCOM3 = 1

Sınıf B için; LCOM3 = 1

Değerlendirme:

- LCOM3 ölçütünün minimum değeri "1" ve maximum değeri metot sayısı ile değişiyor. LCOM3 normalize edilmiş bir ölçüt değildir.
- Metot çağrımları göz önüne alınmamıştır.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- LCOM3 ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak LCOM3 hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize LCOM3 ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da LCOM3 ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.

4.2.2. LCOM4

Hitz ve Montazeri tarafından 1995 yılında tanımlanan ölçüttür (Hitz ve Montazeri, 1995). LCOM3 ölçütünden yola çıkılarak önerilmiştir.

Tanım :

X bir sınıf olsun. I_x, X sınıfının üye alanlarının kümesi olsun. M_x, X sınıfının metotlarının kümesi olsun. Yönsüz bir grafik $G_x(V, E)$ göz önüne alalım, uçları sınıfın metotları olsun.

$V = M_x$ (4.5)

$$E = \left\{ < m, n > \in V \times V \middle| \exists i \in I_X : (\text{m accesses i}) \wedge (n \text{ accesses i}) \vee (m \text{ calls n}) \vee (n \text{ calls m}) \right\} \quad (4.6)$$

LCOM4, G_x in bağlı bileşenlerinin sayısı olarak tanımlanıyor.

Diğer bir ifadeyle, LCOM3'e benzer. Yalnız burada G grafiğinde eğer metotlardan biri diğerini çağırıyorsa o zaman da aralarında bir kenar oluşturuluyor.

LCOM4 metriği ters uyum ölçümüdür. Yüksek LCOM4 değeri düşük uyumu gösterirken düşük LCOM4 değeri yüksek uyumu gösterir.

LCOM4 Ölçütü Hesaplaması:

Örnek 1:

7) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

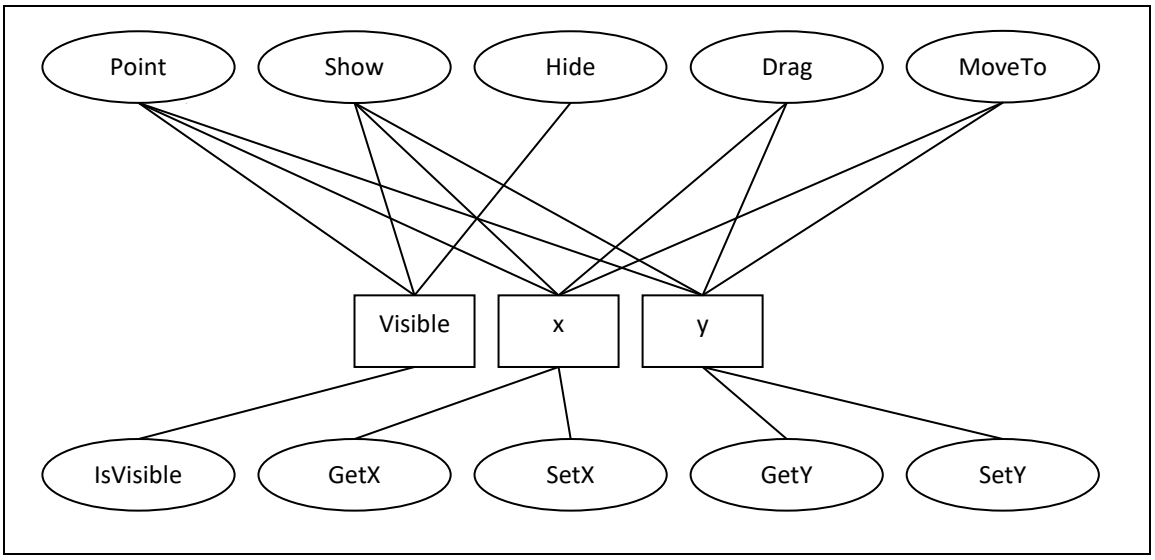
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

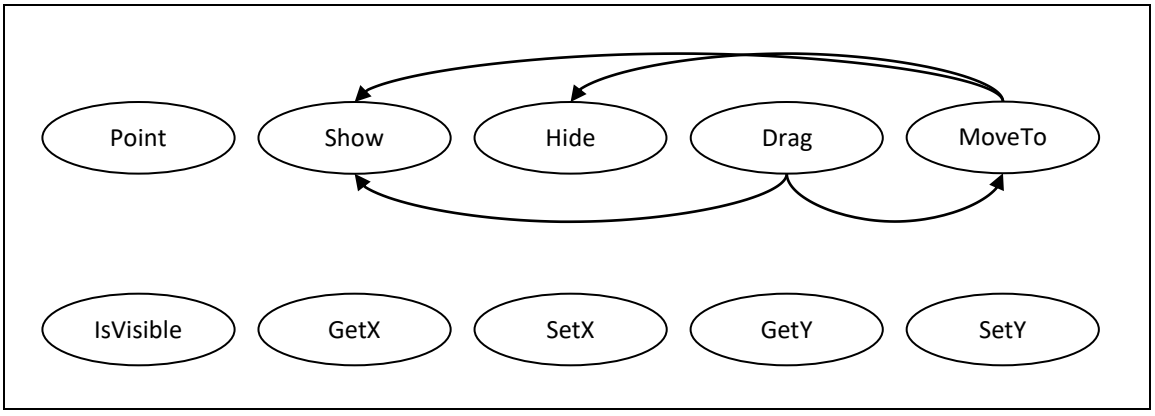
I_4: {x, y} I_5: {x, y} I_6: {Visible}

M_7: GetX() M_8:SetX() M_9: GetY() M_{10}: SetY()

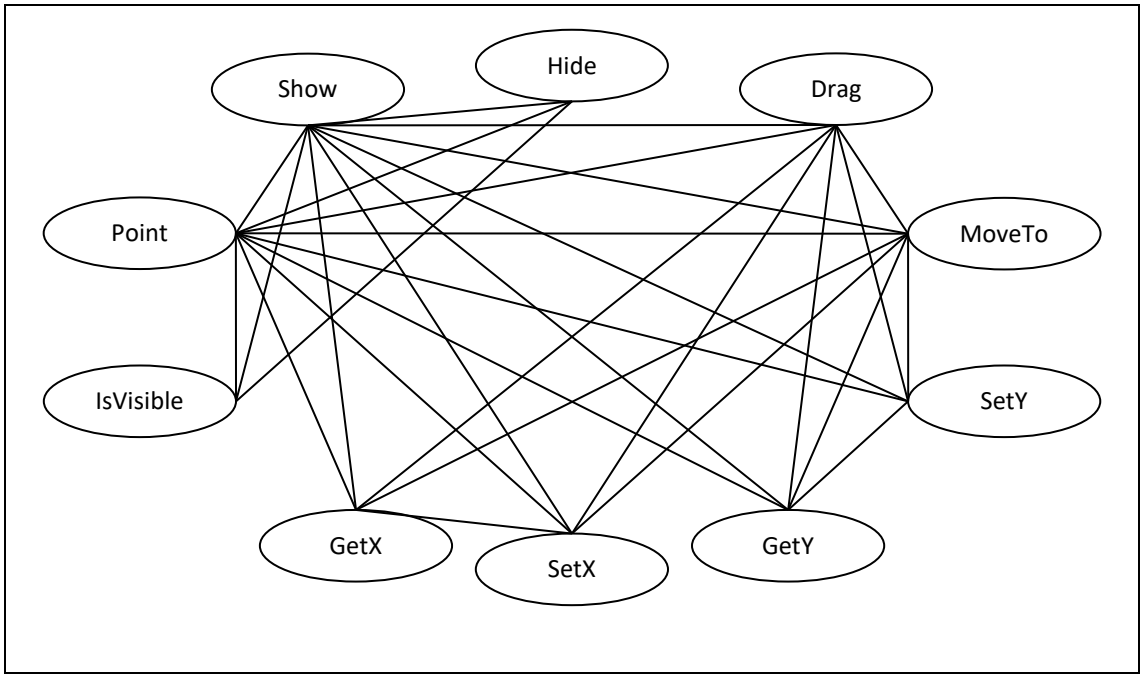
I_7: {x} I_8: {x} I_9: {y} I_{10}: {y}



Şekil 4.2.8 LCOM4 (1) şıkkına göre metot-üye alan ilişkisi grafiği



Şekil 4.2.9 LCOM4 (1) şıkkına göre metot çağrımı ilişkisi grafiği



Şekil 4.2.10 LCOM4 (1) şıkkına göre Gx grafiği

LCOM4= 1

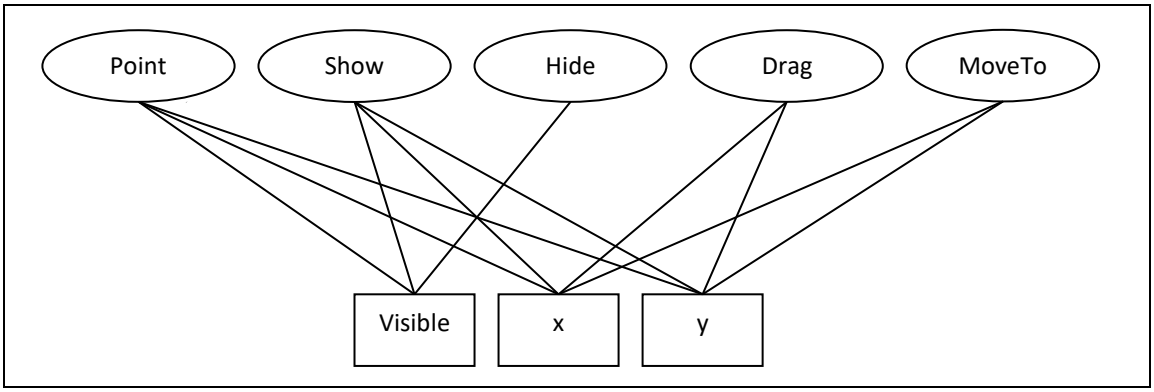
8) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

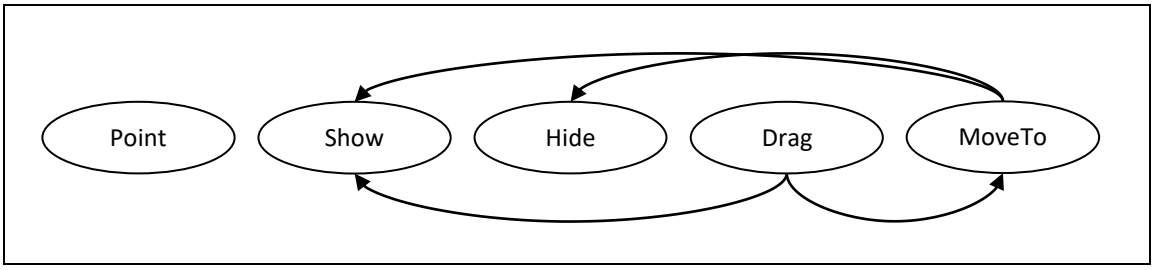
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

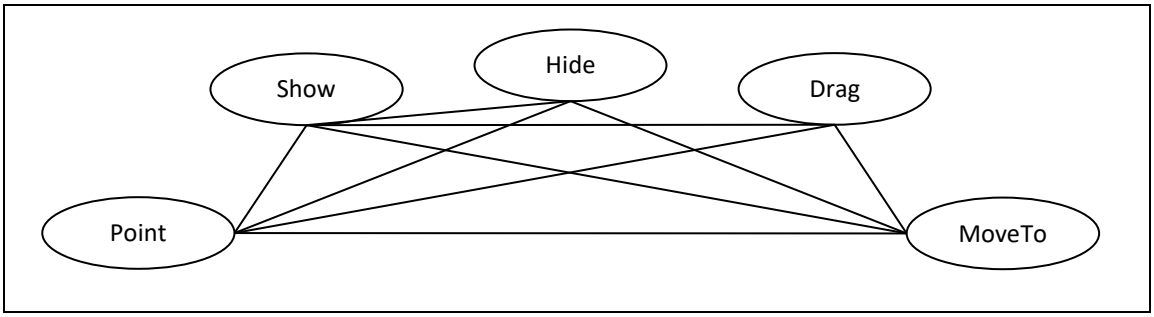
I_4: {x, y} I_5: {x, y}



Şekil 4.2.11 LCOM4 (2) şıkkına göre metot üye alan ilişkisi



Şekil 4.2.12 LCOM4 (2) şıkkına göre metot çağrımı ilişki grafiği



Şekil 4.2.13 LCOM4 (2) şıkkına göre Gx grafiği

LCOM4 = 1

9) Constructor metotları hesaplamaya alınmayacak, erişim metotlar hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

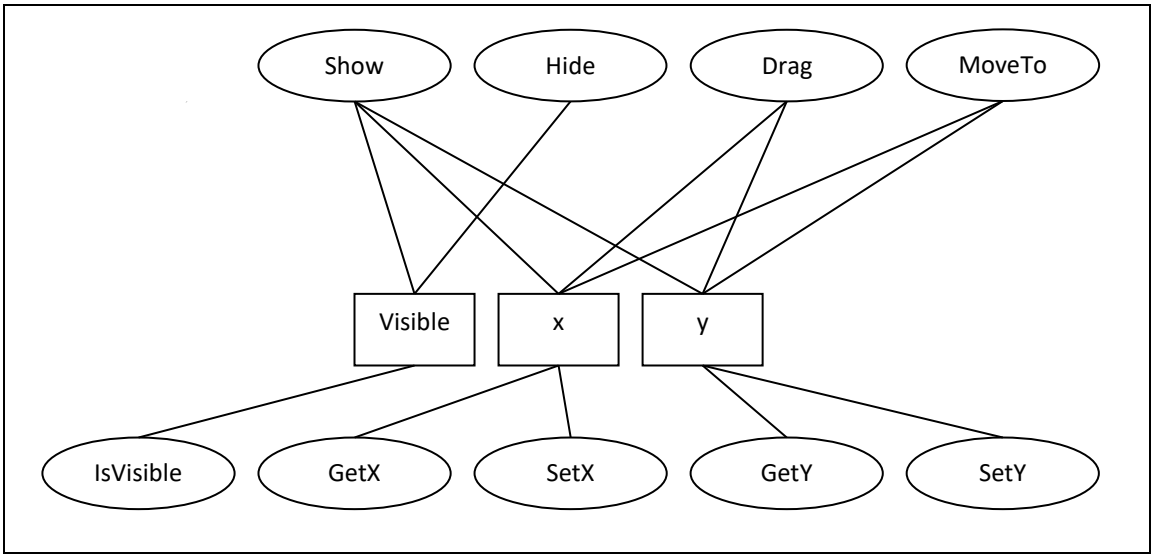
I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

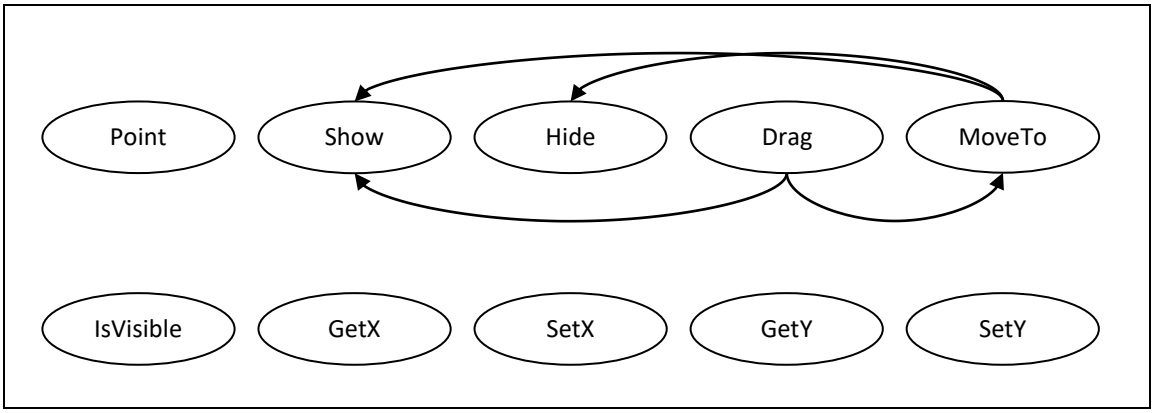
I_5: {x, y} I_6: {Visible} I_7: {x}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}

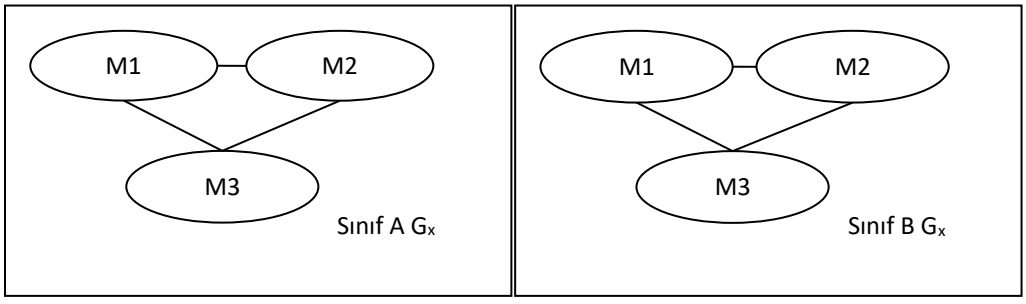


Şekil 4.2.14 LCOM4 (3) şıkkına göre metot çağrımı ilişki grafiği



Şekil 4.2.15 LCOM4 (3) şıkkına göre metot çağrımı ilişki grafiği

Örnek 2:



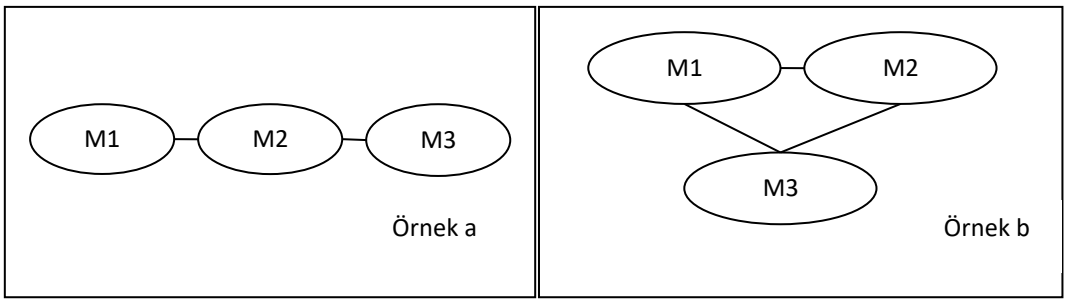
Şekil 4.21 LCOM4 Sınıf A ve Sınıf B'ye ait Gx grafiği

Sınıf A için; LCOM4 = 1

Sınıf B için; LCOM4= 1

Değerlendirme:

- LCOM4 ölçütünün minimum değeri "1" ve maximum değeri metot sayısı ile değişiyor. LCOM4 normalize edilmiş bir ölçüt değildir.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- Örnek 1 için yapılan LCOM4 ölçümü sonucunda, erişim metotlarının ve constructor metotların göz önüne alınmasının *uyumu* etkilemediği görülmüştür.
- LCOM4 ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak LCOM4 hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize LCOM4 ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da LCOM4 ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.
- Yönsüz bir grafikte 1 tane bağlı bileşen bulunduğu durumda her zaman LCOM4 =1 değeri elde ediliyor. LCOM4 değeri 1 çıkan sonuçlarda farklı *uyum* özelliği taşımak mümkün. Bu da LCOM4 ölçütünün sınıfın *uyum* özelliğini iyi ölçemediğini gösteriyor. Bu nedenle LCOM4=1 için LCOM4 ölçütü gerçek *uyum* özelliğini yansıtmıyor.
  Şöyle ki: 2 örneği ele alırsak;



Yukarıdaki 2 örnekte de LCOM4 =1 dir. Ancak Örnek b. de metot ilişkileri daha çok olmasına rağmen aynı LCOM4 değeri elde edilmiş. Kenar sayısı ne kadar fazlaysa uyum o kadar yükselir. Bu durumu göz önüne alarak Hitz ve Montazeri, Co ölçütünü önermişlerdir.

4.3. Co

Hitz ve Montazeri tarafından 1995 yılında tanımlanan ölçüttür (Hitz ve Montazeri, 1995). LCOM4 ölçütünün bir varyasyonudur. LCOM4 =1 değerine sahip sınıfların uyum özelliğini ayırt etmek üzere bağlı bileşenin *kenar sayısı* hesaba katılarak Co ölçütü önerilmiştir.

Tanım :

E ve V sırasıyla c sınıfının Gx bağlı grafiğinin kenarları ve köşeleri olsun.

LCOM4 = 1 olduğu durumda (*Gx* sadece 1 bağlı bileşen içersin), kenar sayısı E şu değerler arasında değişir: $|V| - 1$ (minimum uyum) ve $|V|.(|V| - 1)/2$ (maximum uyum). Co ölçütü bağlı bileşen 1 tane olduğu durumlarda uygulanabilen bir ölçüttür. Co ölçütü şu şekilde tanımlanıyor.

$$Co = 2.\frac{|E| - (|V| - 1)}{(|V| - 1).(|V| - 2)} \qquad (4.7)$$

Co Ölçütü Hesaplaması:

Örnek 1:

10) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

1 adet bağlı bileşen var. Ölçüt uygulanabilir. E = 31 ve V = 10.

$$Co = 2.\frac{|31| - (|10| - 1)}{(|10| - 1).(|10| - 2)} = 2.\frac{22}{9.8} = \frac{11}{18} = 0.61$$

11) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

1 adet bağlı bileşen var. Ölçüt uygulanabilir. E = 9 ve V = 5.

$$Co = 2.\frac{|9| - (|5| - 1)}{(|5| - 1).(|5| - 2)} = 2.\frac{5}{4.3} = \frac{5}{6} = 0.83$$

12) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

1 adet bağlı bileşen var. Ölçüt uygulanabilir. E = 20 ve V = 9.

$$Co = 2.\frac{|20| - (|9| - 1)}{(|9| - 1).(|9| - 2)} = 2.\frac{12}{8.7} = 0.42$$

Örnek 2:

Sınıf A için;

1 adet bağlı bileşen var. Ölçüt uygulanabilir. E = 3 ve V = 3.

$$Co = 2.\frac{|3|-(|3|-1)}{(|3|-1).(|3|-2)} = 2.\frac{1}{2.1} = 1$$

Sınıf B için;

1 adet bağlı bileşen var. Ölçüt uygulanabilir. E = 3 ve V = 3.

$$Co = 2.\frac{|3|-(|3|-1)}{(|3|-1).(|3|-2)} = 2.\frac{1}{2.1} = 1$$

Değerlendirme:

- Co ölçütü [0, 1] arasında değerler alıyor. 0 değeri düşük uyumu 1 değeri yüksek uyumu gösteriyor. Normalize edilmiştir.
- Grafikte sadece 1 tane bağlı bileşen olduğu durumlarda uygulanabiliyor.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- Co ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 1 için yapılan Co ölçümü sonucunda erişim metotlarının, *uyumu*, yapay olarak düşürdüğü, constructor metotların *uyumu* yapay olarak arttırdığı görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak Co hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize Co ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da Co ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.

4.4. Bieman ve Kang tarafından Önerilen Ölçütler

Bieman ve Kang 1995 yılında Geniş C++ Sistemleri için mantıklı sayılabilecek ve Chidamber ve Kemerer ölçütleri temeline dayanan (Briand vd., 1998) iki yeni ölçüt önermiş ve bu ölçütleri sınıf uyumu ve tekrar kullanım için değerlendirmişlerdir (Briand vd., 1995).Çalışmaları sonucu görülmüştür ki, bir sınıf ne kadar sık miras yoluyla tekrar kullanılıyorsa o kadar az uyuma sahiptir.

Bieman ve Kang'e göre; Bir sınıfın bileşenleri sınıfta tanımlanan üye alanlar ve metotlar ve buna ilave miras alınanlardır. Bir metot ve üye alan şu yolla ilişkilidir; eğer üye alan metot tarafından kullanılıyorsa. İki metot ilişkilidir; eğer her iki metot da üye alanı kullanıyorsa. Sınıf uyumu, sınıftaki bağlı metotların sayısı olarak tanımlanabilir.

Burada metot çiftlerinin doğrudan veya dolaylı yoldan bağlanmalarına dayalı iki sınıf uyum ölçütü tanımlanıyor;

- TCC (Tight Class Cohesion) (doğrudan)
- LCC (Loose Class Cohesion) (doğrudan + dolaylı)

Bu iki ölçüt de "duyarlı" (Bir sınıftaki uyum değişikliğinin ölçüm sonucunu etkilemesi) olarak belirtilmişlerdir.

Bieman ve Kang'e göre metotlar 2 yolla bağlıdır (Bieman ve Kang, 1995);

- **MIV ilişkileri:** paylaşılan üye alanlar yoluyla olan metotlar arasındaki haberleşmeyi içerir.
- **Call ilişkileri:** bir metottan diğerine doğrudan veya dolaylı yoldan mesajların gönderilmesini içerir.

Constructor'lar genellikle bir sınıftaki bütün üye alanlara erişir ve böylece üye alanlar bütün diğer metotlarla virtual olarak paylaşır. Bu nedenle çalışmalarında constructor metotları çıkarmışlardır. Destructor metotları da kullanıma almamışlardır. Bieman ve Kang ölçüt tanımlamalarından önce şu belirtimleri yapmışlardır:

Eğer, *üye alan, M* metodunda veri belirtimi olarak görünüyorsa o zaman bu üye alan metot *M* tarafından *doğrudan kullanılıyor* demektir. *Üye alan M* metodu ile aynı sınıfta veya sınıfın ata sınıfında tanımlanabilir.

Eğer *call graph*'ta *M* metodu *M´* metodunun öncesinde geliyorsa *M´* metodu *M* metodu tarafından *doğrudan çağrılıyor* demektir. Eğer *call graph*'ta *M* metodundan *M´* metoduna bir yol varsa *M´* metodu *M* metodu tarafından *dolaylı yoldan çağrılıyor* demektir.

Eğer, *üye alan*, *M* metodu tarafından doğrudan veya dolaylı yoldan çağrılan *M´* metodu tarafından doğrudan kullanılıyorsa o zaman bu *üye alan* metot *M* tarafından *dolaylı yoldan kullanılıyor* demektir.

Eğer iki metot arasında bir veya daha fazla ortak *üye alan* varsa bu iki metot *doğrudan ilişkili*dir. İki metot diğer doğrudan ilişkili metotlar üzerinden bağlıysa o zaman bu iki metot *dolaylı ilişkili*dir.

NP(C): bir sınıftaki mümkün olan maximum doğrudan veya dolaylı bağlantı sayısı. Eğer bir sınıfta N metot varsa o zaman NP(C);

$$NP(C) = \frac{N * (N - 1)}{2} \qquad (4.8)$$

NDC(C): doğrudan bağlantı sayısı

NIC(C): dolaylı bağlantı sayısı

4.5. TCC

Tanım :

Bir sınıftaki doğrudan bağlı metot çifti sayısının, NDC(C), bir sınıftaki mümkün olan maximum doğrudan veya dolaylı bağlantı sayısına, NP(C), oranı olarak bulunur..

$$TCC = \frac{NDC(C)}{NP(C)} \qquad (4.9)$$

TCC Ölçütü Hesaplaması:

Örnek 1:

13) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

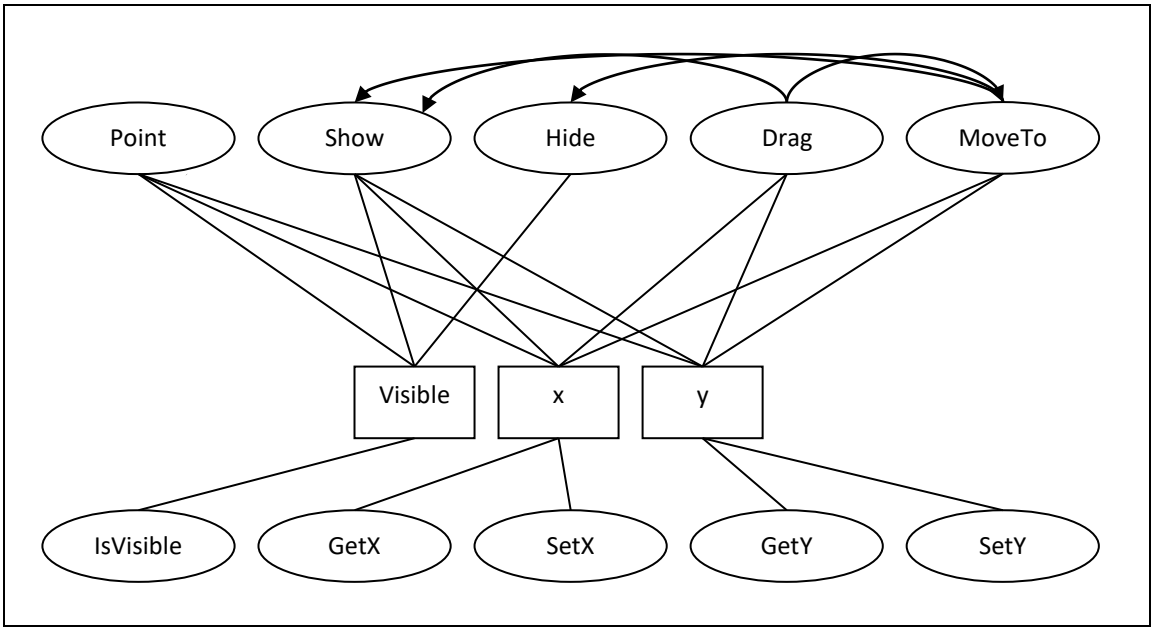
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

I_4: {x, y} I_5: {x, y} I_6: {Visible}

M_7: GetX() M_8:SetX() M_9: GetY() M_{10}: SetY()

I_7: {x} I_8: {x} I_9: {y} I_{10}: {y}



Şekil 4.5.1 TCC (1) şıkkına göre metot- üye alan ve metot çağrımı ilişkisi

Doğrudan bağlı metot çiftleri:

{M1, M2}, {M1, M3}, {M1, M4}, {M1, M5}, {M1, M6}, {M1, M7}, {M1, M8}, {M1, M9}, {M1, M10}, {M2, M3}, {M2, M4}, {M2, M5}, {M2, M6}, {M2, M7}, {M2, M8}, {M2, M9}, {M2, M10}, {M3, M6}, {M4, M5}, {M4, M7}, {M4, M8}, {M4, M9}, {M4, M10}, {M5, M7}, {M5, M8}, {M5, M9}, {M5, M10}, {M7, M8}, {M9, M10}

NDC(C) = 29

Dolaylı bağlı metot çiftleri: {M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9}, {M_8, M_{10}}

NIC(C) = 16

$$NP(C) = \frac{10*(10-1)}{2} = 45$$

$$TCC = \frac{29}{45} = 0.64$$

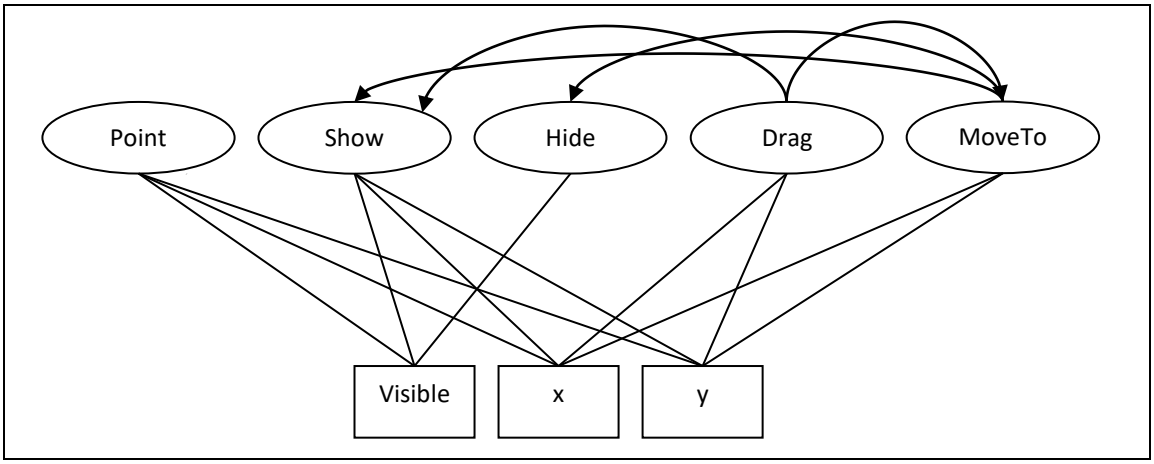
14) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

I_4: {x, y} I_5: {x, y}



Şekil 4.5.2 TCC (2) şıkkına göre metot-üye alan ve metot çağrımı ilişkisi

Doğrudan bağlı metot çiftleri:

{M_1, M_2}, {M_1, M_3}, {M_1, M_4}, {M_1, M_5}, {M_2, M_3}, {M_2, M_4}, {M_2, M_5}, {M_4, M_5}

NDC(C) = 8

Dolaylı bağlı metot çiftleri: {M_3, M_4}, {M_3, M_5}

NIC(C) = 2

$$NP(C) = \frac{5*(5-1)}{2} = 10$$

$$TCC = \frac{8}{10} = 0.8$$

15) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

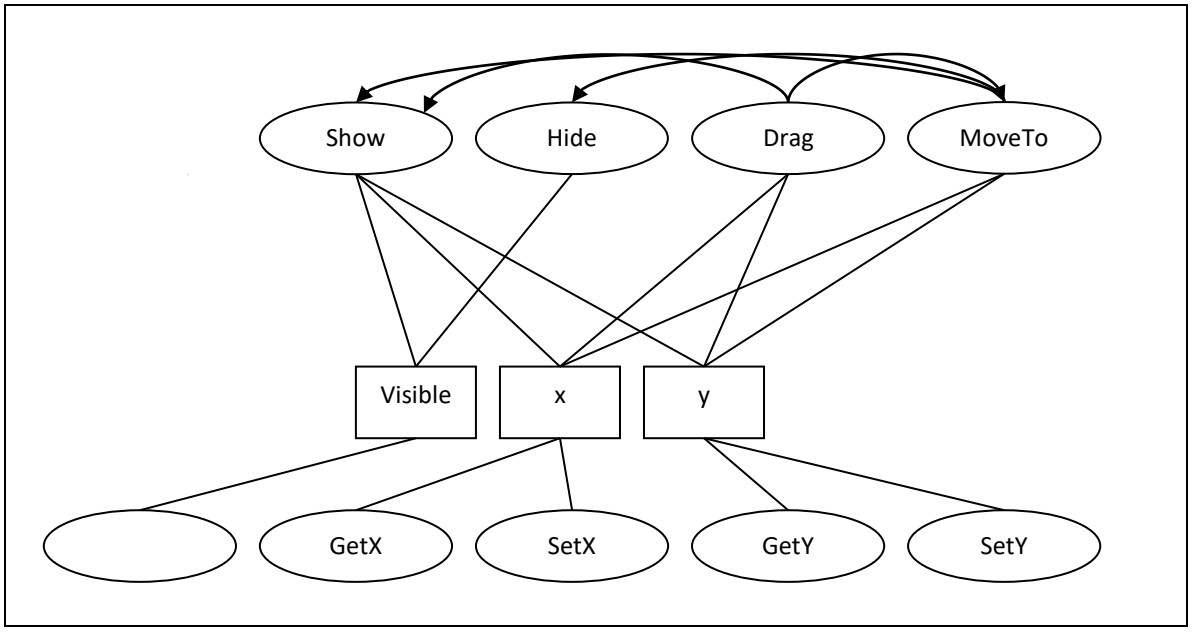
I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

I_5: {x, y} I_6: {Visible} I_7: {x}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}



Şekil 4.5.3 TCC (3) şıkkına göre metot-üye alan ve metot çağrımı ilişkisi

Doğrudan bağlı metot çiftleri:

{M_2, M_3}, {M_2, M_4}, {M_2, M_5}, {M_2, M_6}, {M_2, M_7}, {M_2, M_8}, {M_2, M_9}, {M_2, M_{10}}, {M_3, M_6}, {M_4, M_5}, {M_4, M_7}, {M_4, M_8}, {M_4, M_9}, {M_4, M_{10}}, {M_5, M_7}, {M_5, M_8}, {M_5, M_9}, {M_5, M_{10}}, {M_7, M_8}, {M_9, M_{10}}

NDC(C) = 20

Dolaylı bağlı metot çiftleri: {M_3, M_4}, {M_3, M_5}, {M_3, M_7}, {M_3, M_8}, {M_3, M_9}, {M_3, M_{10}}, {M_4, M_6}, {M_5, M_6}, {M_6, M_7}, {M_6, M_8}, {M_6, M_9}, {M_6, M_{10}}, {M_7, M_9}, {M_7, M_{10}}, {M_8, M_9}, {M_8, M_{10}}

NIC(C) = 16

$$NP(C) = \frac{9*(9-1)}{2} = 36$$

$$TCC = \frac{20}{36} = 0.55$$

Örnek 2:

Sınıf A için:

Doğrudan bağlı metot çiftleri: {M_1, M_2}, {M_1, M_3}, {M_2, M_3}

NDC(C)=3

Dolaylı bağlı metot çiftleri: -

NIC(C) = 0

$$NP(C) = \frac{3*(3-1)}{2} = 3$$

$$TCC = \frac{3}{3} = 1$$

Sınıf B için:

Doğrudan bağlı metot çiftleri: {M_1, M_2}, {M_1, M_3}, {M_2, M_3}

NDC(C)= 3

Dolaylı bağlı metot çiftleri: -

NIC(C) = 0

$$NP(C) = \frac{3*(3-1)}{2} = 3$$

$$TCC = \frac{3}{3} = 1$$

Değerlendirme:

- TCC ölçütü [0, 1] arasında değerler alıyor. 0 değeri düşük uyumu 1 değeri yüksek uyumu gösteriyor. Normalize edilmiştir.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- Bieman ve Kang'in Constructor'ları ve Destructor'ları hesaba katmamayı önermişlerdir.
- Örnek 1 için yapılan TCC ölçümü sonucunda erişim metotlarının, *uyumu*, yapay olarak düşürdüğü, constructor metotların *uyumu* yapay olarak arttırdığı görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak TCC hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize TCC ölçütünün sezgilerle paralellik

göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da TCC ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.

4.6. LCC

Tanım :

Bir sınıftaki doğrudan bağlı metot çifti sayısı, NDC(C), ve dolaylı bağlı metot çifti sayısının, NIC(C), toplamının, bir sınıftaki mümkün olan maximum doğrudan veya dolaylı bağlantı sayısına, NP(C), oranı olarak bulunur.

$$LCC = \frac{NDC(C) + NIC(C)}{NP(C)} \quad (4.10)$$

LCC Ölçütü Hesaplaması:

Örnek 1:

16) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

NDC(C) = 29, NIC(C) = 16, NP(C) = 45 (Bu değerler TCC ölçütünde hesaplanmıştır)

$$LCC = \frac{29+16}{45} = 1$$

17) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

NDC(C) = 8, NIC(C) = 2, NP(C) = 10 (Bu değerler TCC ölçütünde hesaplanmıştır)

$$LCC = \frac{8+2}{10} = 1$$

18) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

NDC(C) = 20, NIC(C) = 16, NP(C) = 36 (Bu değerler TCC ölçütünde hesaplanmıştır)

$$LCC = \frac{20+16}{36} = 1$$

Örnek 2:

Sınıf A için: *Sınıf B için:*

$$LCC = \frac{3+0}{3} = 1 \qquad LCC = \frac{3+0}{3} = 1$$

Değerlendirme:

- LCC ölçütü [0, 1] arasında değerler alıyor. 0 değeri düşük uyumu 1 değeri yüksek uyumu gösteriyor. Normalize edilmiştir.
- Metot çağrımları, dolaylı yoldan üye alan kullanımına bakmak amacıyla kullanılmış, metotların birbirini çağırma anlamında metot ilişkisi göz önüne alınmamıştır.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- Bieman ve Kang'in Constructor'ları ve Destructor'ları hesaba katmamayı önermişlerdir.
- Örnek 1 için yapılan LCC ölçümü sonucunda, erişim metotlarının ve constructor metotların göz önüne alınmasının *uyumu* etkilemediği görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak LCC hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize LCC ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğine sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da LCC ölçütünün uyumu çok da iyi ayırt edemediğini gösteriyor.

4.7. Henderson-Sellors tarafından Önerilen Ölçütler

4.8.4. LCOM5

Henderson ve Sellors tarafından 1996 yılında tanımlanan ölçüttür. Chidamber ve Kemerer'in LCOM ölçütünden esinlenerek önerilmiştir.

Tanım :

M_i metot kümesi (i=1, … , m) ve bu metotların eriştiği üye alan kümesi de A_j (j=1, … , a) olsun. $\mu(A_j)$, A_j üye alanına referans eden metot sayısı olsun. LCOM5 şu şekilde tanımlanıyor:

$$LCOM5 = \frac{\frac{1}{a}\sum_{j=1}^{a}\mu(A_j) - m}{1-m} \qquad (4.11)$$

LCOM5 Ölçütü Hesaplaması:

Örnek 1:

19) Erişim metotları hesaplamaya alınacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

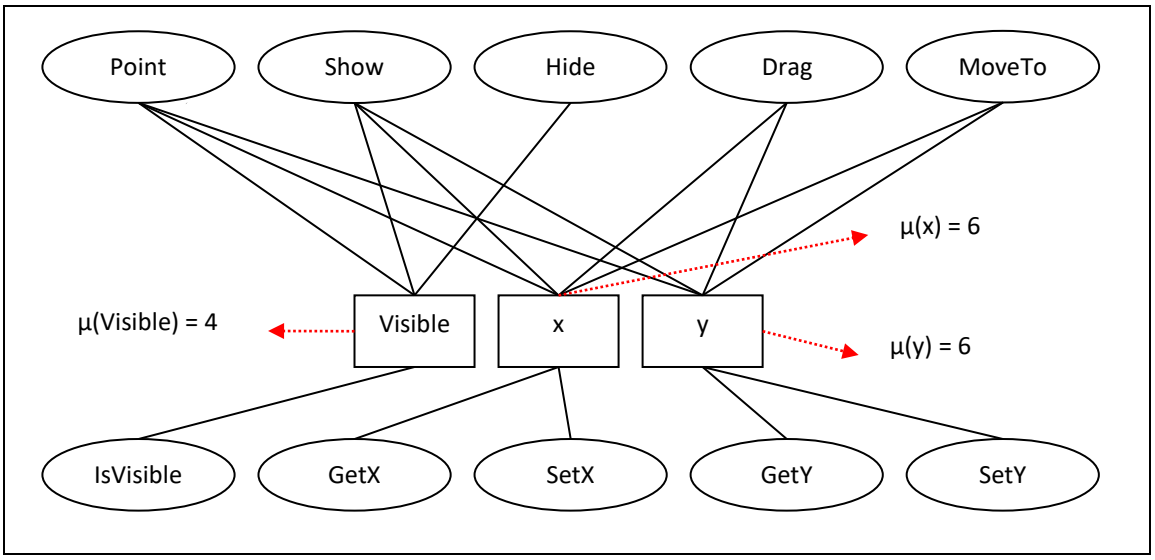
I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo() M_6: IsVisible()

I_4: {x, y} I_5: {x, y} I_6: {Visible}

M_7: GetX() M_8:SetX() M_9: GetY() M_{10}: SetY()

I_7: {x} I_8: {x} I_9: {y} I_{10}: {y}



Şekil 4.7.1 LCOM5 (1) şıkkına göre metot-üye alan ilişkisi ve μ değerleri

Üye alan sayısı a = 3 ve metot sayısı m = 10.

$$LCOM5 = \frac{\frac{1}{a}[\mu(Visible) + \mu(x) + \mu(y)] - m}{1 - m}$$

$$LCOM5 = \frac{\frac{1}{3}[4 + 6 + 6] - 10}{1 - 10} = 0.51$$

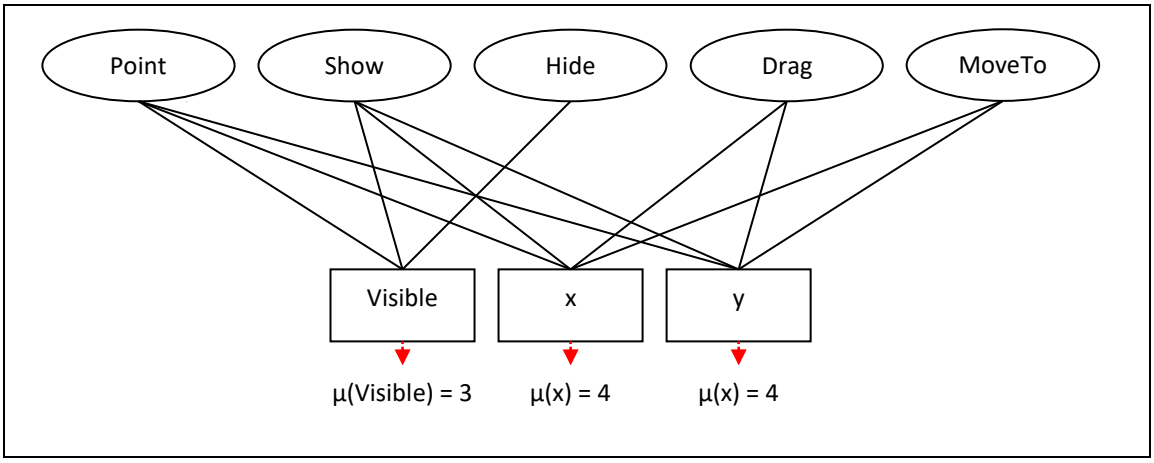
20) Erişim metotları hesaplamaya alınmayacak, constructor metotlar hesaplamaya alınacak

M_1: Point() M_2: Show() M_3: Hide()

I_1: {Visible, x, y} I_2: {Visible, x, y} I_3: {Visible}

M_4: Drag() M_5: MoveTo()

I_4: {x, y} I_5: {x, y}



Şekil 4.7.2 LCOM5 (2) şıkkına göre metot-üye alan ilişkisi ve μ değerleri

Üye alan sayısı a = 3 ve metot sayısı m = 5.

$$LCOM5 = \frac{\frac{1}{3}[3+4+4]-5}{1-5} = 0.33$$

21) Constructor metotlar hesaplamaya alınmayacak, erişim metotları hesaplamaya alınacak

M_2: Show() M_3: Hide() M_4: Drag()

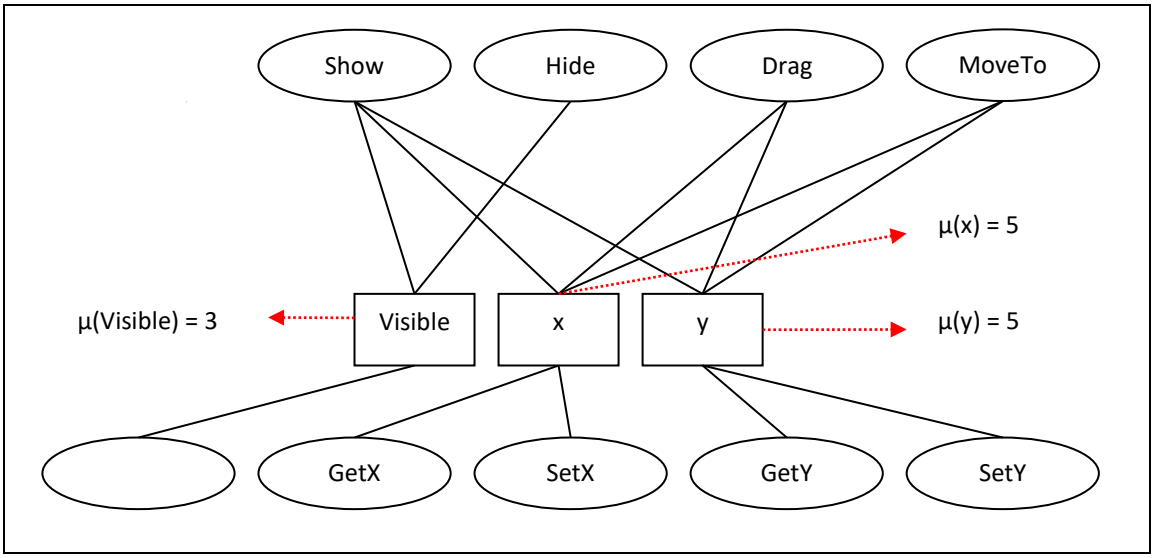
I_2: {Visible, x, y} I_3: {Visible} I_4: {x, y}

M_5: MoveTo() M_6: IsVisible() M_7: GetX()

I_5: {x, y} I_6: {Visible} I_7: {x,}

M_8:SetX() M_9: GetY() M_{10}: SetY()

I_8: {x} I_9: {y} I_{10}: {y}

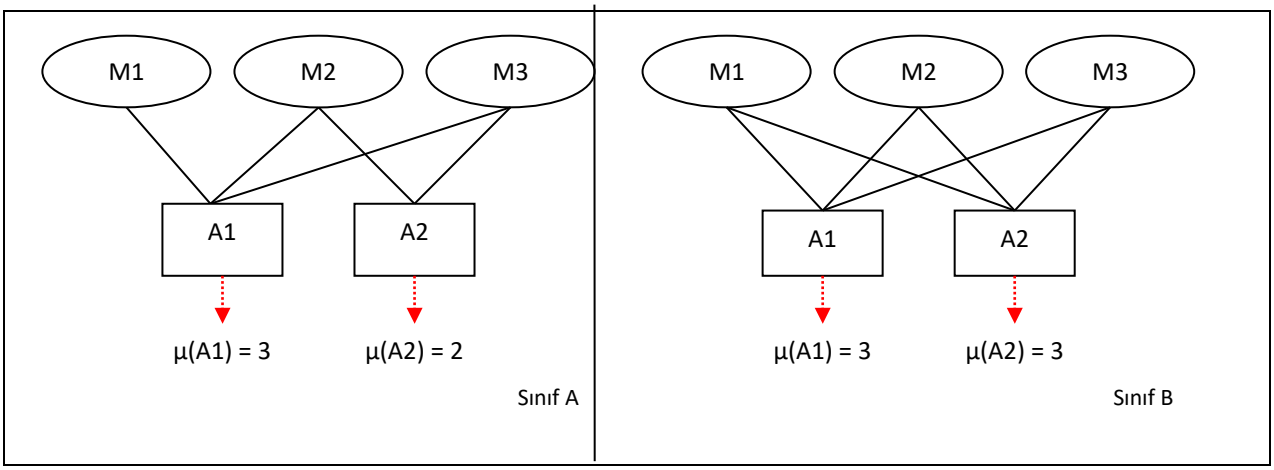


Şekil 4.7.3 LCOM5 (3) şıkkına göre metot-üye alan ilişkisi ve μ değerleri

Üye alan sayısı a = 3 ve metot sayısı m = 9.

$$LCOM5 = \frac{\frac{1}{3}[3+5+5]-9}{1-9} = 0.58$$

Örnek 2:



Şekil 4.7.4 LCOM5 Sınıf A ve Sınıf B metot-üye alan ilişkisi ve μ değerleri

Sınıf A için;

Üye alan sayısı a = 2 ve metot sayısı m = 3.

$$LCOM5 = \frac{\frac{1}{a}[\mu(A1)+\mu(A2)]-m}{1-m}$$

$$LCOM5 = \frac{\frac{1}{2}[3+2]-3}{1-3} = 0.25$$

Sınıf B için;

Üye alan sayısı a = 2 ve metot sayısı m = 3.

$$LCOM5 = \frac{\frac{1}{2}[3+3]-3}{1-3} = 0$$

Değerlendirme:

- LCOM5 ölçütü, bir sınıfın her bir üye alanı en az 1 metot tarafından referans ediliyor varsayımı yapıldığı durumda 0 ve 1 arasında normalize edilmiş oluyor. Eğer bu varsayım çıkarılırsa bu durumda 0 ve 2 arasında normalize edilmiş oluyor. Halen bir normalize durum var fakat maximum değer 2 ancak m = 2 olduğunda geçerlidir. Bu durumu şu örnekle; LCOM5 hiçbir metot tarafından referans edilen üye alan olmadığı durumda (LCOM5= m/m-1) maximum değer 2 alıyor. m>2 olduğu durumlarda maximum değer küçülüyor. Bu da bir anomali durumdur.
- Formülü incelediğimizde bazı noktalarda LCOM5 ölçütünün istenmeyen sonuçlar verebileceğini görüyoruz. Şöyle ki, eğer ölçülen sınıfta hiç üye alan yoksa veya sadece 1 tane metot varsa o zaman ölçüt sonsuz bir değere çıkıyor. Bu durum da ölçme teorisi kuralına ters düşen bir durumdur. İyi bir ölçüt her zaman mantıklı sonuçlar veren ve böyle sonuçlar alma ihtimali olmayan ölçüttür.
- Metot çağrımları göz önüne alınmamıştır.
- Dolaylı bağlantılar göz önüne alınmamıştır.
- Teorik olarak ve deneysel olarak geçerlenmemiştir.
- LCOM5 ölçütünde hesaplama yaparken miras özelliğinin ve özel metotların kullanımına yönelik herhangi bir belirtim yapılmamıştır.
- Örnek 1 için yapılan LCOM5 ölçümü sonucunda erişim metotlarının, *uyumu*, yapay olarak düşürdüğü, constructor metotların *uyumu* yapay olarak arttırdığı görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan daha iyi çıkmalıdır. LCOM5 ölçütünün sezgilerle paralel sonuçlar verdiği görülmektedir.

4.8. Lee et al. tarafından Önerilen Ölçütler

5.6.4. ICH

ICH (information-flow based cohesion) ölçütü Lee tarafından 1995 yılında önerilmiş bir ölçüttür (Briand vd., 1998). Lee et al. bir sınıftaki metot çağrımları boyunca oluşan bilgi akışına dayalı bir uyum ölçütü kümesi önermiştir.

Tanım :

Bir *c* sınıfında implement edilen (yeni tanımlanan veya miras alınıp override edilen) metot *m* için, *m* metodunun uyumu *c* sınıfında implement edilen diğer metotların çağrılma sayısının çağrılan metotların parametre sayısı ile ağırlıklandırılması olarak ölçülür. Çağrılan metot ne kadar çok parametreye sahipse o kadar çok bilgi akışı vardır ve çağrılan ve çağıran metot arasında o kadar çok güçlü bir bağ vardır demektir. Bir sınıfın uyumu, metotlarının uyumunun toplamı olarak ölçülür. Bir sınıf kümesinin uyumu ise kümedeki sınıfların uyumunun toplamı olarak ölçülür. Formül olarak bu ölçütler şu şekilde ifade edilir.

Metot *m* için uyum ölçümü: $ICH(m) = \sum_{m' \in M_{NEW}(c) \cup M_{OVR}(c)} (1 + |Par(m')|).NPI(m, m')$ (4.12)

Sınıf *c* için uyum ölçümü: $ICH(c) = \sum_{m \in M_I(c)} ICH(m)$ (4.13)

Sınıf kümesi *SS* için uyum ölçümü: $ICH(SS) = \sum_{c \in SS} ICH(c)$ (4.14)

$M_{NEW}(c)$: yeni tanımlanmış metotlar kümesi

$M_{OVR}(c)$: override edilmiş metotlar kümesi

$Par(m')$ = m' metodunun parametre kümesi

$NPI(m, m')$ = m metodu tarafından m' metodunun polimorfik çağrılma sayısı

$M_I(c)$ = yeni tanımlanmış veya miras alınıp override edilmiş metotlar kümesi

Değerlendirme:

- ICH ölçütü normalize edilmemiştir.
- Üye alan kullanımı ve/veya paylaşımı göz önüne alınmamıştır.

- Dolaylı bağlantılar göz önüne alınmamıştır.
- Teorik olarak geçerlenmiştir.
- ICH ölçütünde miras özelliğinin hesaba katılacağı belirtilmiş ancak özel metotlarla ilgili herhangi bir şey belirtilmemiştir.
- Örnek 1 için yapılan ICH ölçümü sonucunda erişim metotlarının ve constructor metotların göz önüne alınmasının ölçüm sonuçlarında herhangi bir değişikliğe yol açmadığı görülmüştür.
- Örnek 2 için sezgisel olarak bakıldığında Sınıf B uyumu Sınıf A'dan iyi çıkmalı çünkü Sınıf B'de metotlar daha fazla ortak üye alan kullanmışlar. Ancak ICH hesaplamasına bakıldığında Sınıf A için çıkan değer Sınıf B ile aynı. Bu durum bize ICH ölçütünün sezgilerle paralellik göstermediğini, iki tane farklı uyum özelliğe sahip sınıf için aynı veya yakın değer verebildiğini dolayısıyla da ICH ölçütünün *uyum* özelliğini çok da iyi ayırt edemediğini gösteriyor.
- ICH ölçütü, Birand ve arkadaşları tarafından yapılan çalışmalarda (Briand vd., 2000) deneysel olarak uyumu ölçmek için uygun bir ölçüt olmadığı anlamında geçerlenmiştir. ICH ölçütünün karmaşıklık ölçütü özellikleri taşıdığı yönünde görüşleri vardır. 2 açıdan diğer uyum ölçütlerinden farkı tespit edilmiştir. İlk olarak sadece metot çağrımlarını ele alıyor, sınıfın üye alanlarını hesaba katmıyor. İkinci olarak ICH teorik bir özelliğe sahiptir. ICH bir "katkı" görevi yapıyor. Eğer 2 ilişkisiz fakat yüksek uyum özelliğe sahip sınıf tek bir sınıfta bir araya getirilirse, sınıfın uyumu her iki sınıfın ayrı uyum ölçümlerinin toplamı olacaktır. Bu da birleştirilmiş sınıfın her iki sınıftan daha yüksek uyuma sahip olmasına neden olur. Bu sonuç beklenmeyen bir sonuçtur. Birleştirilmiş sınıfın bir objesi 2 ayrı anlam kavramlarını temsil edeceği için daha az uyum olmalı. Bu sonuç bize ICH ölçütünün pek de uyum ölçütü olmadığını gösteriyor.

5. YENİ ÖLÇÜT COMIAS

COMIAS (Cohesion Method Invocation and Attribute Sharing) uyum ölçütü gerçeklenirken aşağıdaki hususlar dikkate alınmıştır:

- Bir sınıf üye alanlar ve üye metotlardan oluşur. Bu iki önemli elemandan birisinin ölçüt tanımında göz ardı edilmesinin, gerçek uyumun yakalanamamasına neden olacağı düşünülmektedir.
- Bir ölçütün normalize edilmiş olması, bir başka deyişle 0 ile 1 arasında değişen bir değer üretmesi tercih edilir. Normalize edilmiş ölçütlerin yorumlanması daha kolay olacaktır. Buna rağmen literatürde normalize olmayan birçok ölçüt bulunmaktadır.
- Çoğu ölçüt tanımında dolaylı metot ilişkileri göz önüne alınmamıştır. Dolaylı bağlantılar göz

ardı edildiğinde gerçek uyumun yakalanamayacağı düşünülmektedir.

- İyi tanımlanmış bir ölçüt mutlaka sezgilerle paralel sonuçlar vermeli, matematiksel olarak belirsiz ve tanımsız sonuçlar vermemelidir.

Bir sınıfın uyumu COMIAS ölçütü kullanılarak hesaplanırken üç temel adım yürütülmektedir. Birinci adımda üye metotların arasındaki ilişki metot çağrımı yönünden incelenmekte ve ilk grafik (G_{xA}) elde edilmektedir. İkinci adımda ise üye metotlar, üye alanların kullanımı yönünden incelenmekte ve ikinci grafik (G_{xB}) elde edilmektedir. Üçüncü adımda ise ilk iki adımda elde edilen grafikler tek bir grafikte (G_x) birleştirilmektedir. Bu bölümün geri kalanında söz konusu üç adım ayrıntılı olarak incelenecektir.

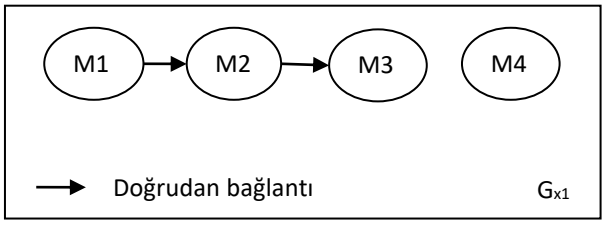
5.1. Metot Çağrımlarının İncelenmesi

Metot çağrımları incelenirken, ilk olarak metotlar arasındaki doğrudan bağlantı ilişkileri incelenir. Eğer bir metot diğerini çağırıyorsa o zaman aralarında çağıran taraftan çağrılan tarafa doğru yönlü bir çizgi oluşturulur. Oluşturulan grafik G_{x1} aşağıdaki şekilde ifade edilebilir:

Yönlü bir grafik $G_{x1}(V, E)$ ve $V = M_x$ ise

$$E = \{< m, n > \in V \times V \,|\, (\text{m calls n})\} \qquad (5.1)$$

Örnek bir G_{x1} grafiği Şekil 6.1'de görülebilir.



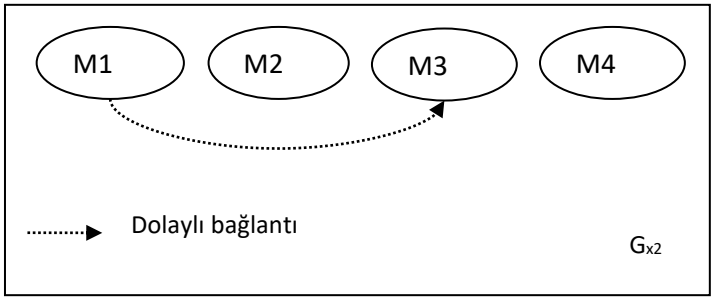
Şekil 5.1.1 Örnek bir doğrudan bağlantı grafiği G_{x1}

Dolaylı bağlantıları bulabilmek için ise G_{x1} grafiğindeki yönlü çizgilerden yararlanılır. Şekil 1 incelendiğinde M1 metodunun M2 metodunu çağırdığı, M2 metodunun da M3 metodunu çağırdığı görülebilir. Dolayısıyla, M1 metodu dolaylı yoldan M3 metodunu çağırmaktadır. M1 metodundan M3 metoduna doğru yönlü bir çizgi oluşturulur. Bu şekilde oluşturulan dolaylı bağlantı grafiği olan G_{x2} aşağıdaki gibi ifade edilebilir:

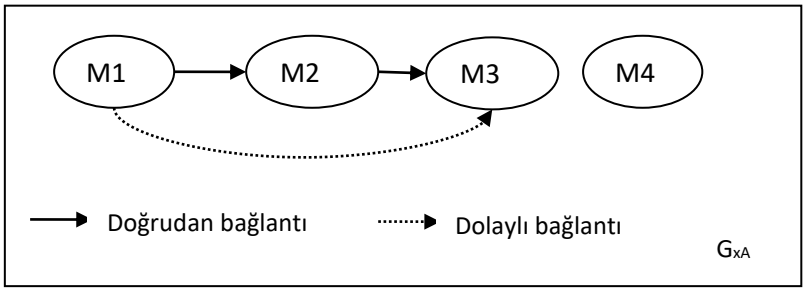
Yönlü bir grafik $G_{x2}(V, E)$ ve $V = M_x$ ise

$$E = \left\{ < m,l > \in V \times V \left| \begin{matrix} (< m,n > \in G_{x1}) \wedge \\ (< n,l > \in G_{x1}) \end{matrix} \right. \right\} \quad (5.2)$$

Örnek G_{x1} grafiğinden elde edilen G_{x2} grafiği Şekil 6.2'de görülebilir.



Şekil 5.1.2 Örnek bir dolaylı bağlantı grafiği G_{x2}

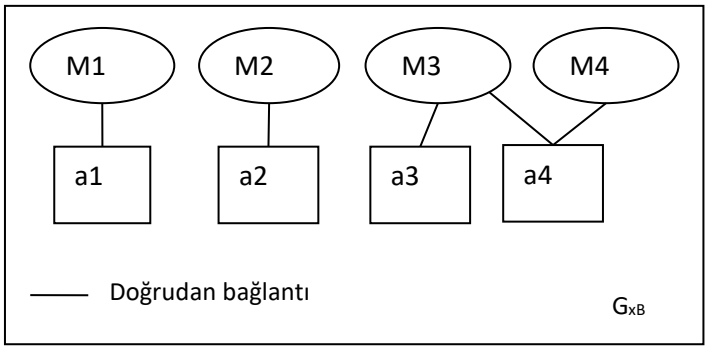


Şekil 5.1.3 Örnek bir metot çağrımı grafiği G_{xA}

Metot çağrımlarının incelenmesinin son adımı olarak, önceki adımlarda elde edilmiş olan Gx1 ve Gx2 grafikleri tek bir grafikte GxA grafiğinde birleştirilerek, doğrudan ve dolaylı yoldan bağlantılı olan tüm metotlar gösterilmiş olunur. Örnek Gx1 ve Gx2 grafiklerinden elde edilen GxA grafiği Şekil 5.1.3'te görülebilir.

5.2. Üye Alan Kullanımlarının İncelenmesi

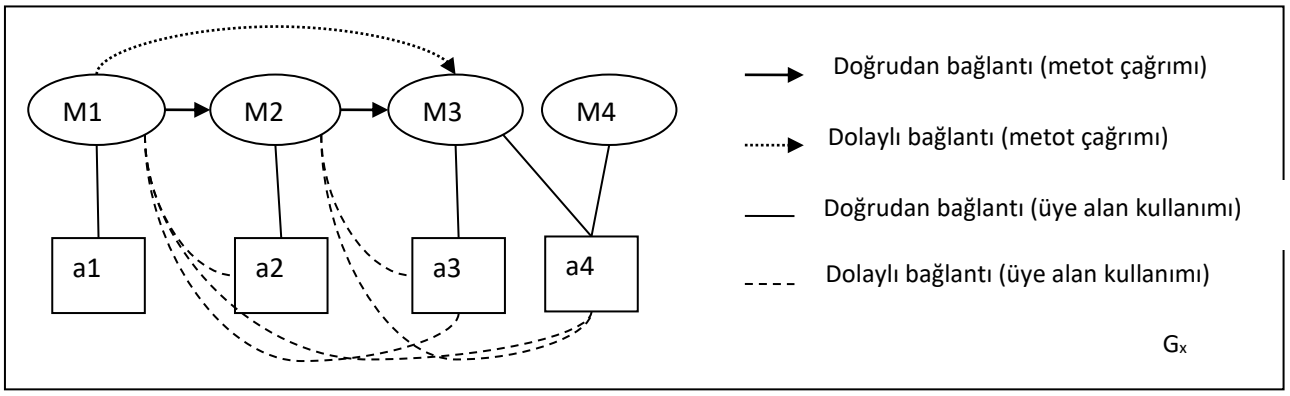
Üye alanların üye metotlar içerisinden kullanımını görebilmek için metot-alan ilişkisini gösteren bir grafik çizilir. Eğer bir metot bir alanı kullanıyorsa o zaman metot ve alan arasında bir çizgi oluşturulur. Şekil 5.4'te bu şekilde elde edilen örnek bir G_{xB} grafiği görülebilir.



Şekil 5.2 Örnek bir doğrudan üye alan kullanımı grafiği G_{xB}

5.3. Metot Çağrımı ve Üye Alan Kullanımı

Önceki adımlarda oluşturulan G_{xA} ve G_{xB} grafikleri bu adımda tek bir grafikte birleştirilerek G_x grafiği elde edilir. Dolaylı metot çağrımlarının incelenmesi ve dolaylı üye alan kullanımlarının tespit edilmesi ile G_x grafiğinin son hali elde edilir. Önceki bölümlerde verilen örnek grafikler kullanılarak elde edilen bir G_x grafiği Şekil 5.5'te görülebilir.



Şekil 5.3 Örnek bir nihai ilişki grafiği G_x

5.4. COMIAS Uyum Ölçütünün Tanımlanması

X bir sınıf, I_x= X sınıfının üye alanlarının kümesi ve M_x = X sınıfının metotlarının kümesi olsun. Yönlü bir grafik $G_x(V, E)$ göz önüne alındığında, iki metot arasındaki metot çağrımı ilişkisi şu şekilde tanımlanacaktır:

$$\vec{Metod}_cag(M_i, M_k) = \begin{cases} 1 \ , eğer < M_i, M_k > \in G_x \\ \quad ya\ da < M_k, M_i > \in G_x \\ 0 \ , diğer\ türlü \end{cases} \qquad (5.3)$$

İki metot arasındaki üye alan paylaşımı ilişkisi, G_x grafiğine bakılarak şu şekilde bulunur:

$$Attr_pay(M_i, M_k) = n \begin{vmatrix} M_i \text{ ve } M_k \text{ metodlarının} \\ \text{paylastıkları attribute sayısı} \end{vmatrix} \quad (5.4)$$

İki metot arasındaki ilişki, metot çağrımı ve üye alan paylaşımlarının toplamı olarak şu şekilde tanımlanır:

İlişki(M_i, M_k) = M_i metodu ile M_k metodu arasındaki ilişki değeri, $M_i \in M_x$ ve $M_k \in M_x$, $i \neq k$ olarak tanımlansın. Bu durumda;

İlişki(M_i, M_k) = Metod_cag(M_i, M_k) + Attr_pay(M_i, M_k)

olarak tanımlanır.

İki metodun ilişkisinden sınıfın toplam ilişki değeri ise şu şekilde bulunur:

$$Toplam_İlişki(X) = \sum_{i=1}^{m-1} \sum_{k=i+1}^{m} \left[İlişki(M_i, M_k) \right] \quad (5.5)$$

$$Toplam_İlişki(X) = \sum_{i=1}^{m-1} \sum_{k=i+1}^{m} \begin{bmatrix} Metod_cag(M_i, M_k) \\ +Attr_pay(M_i, M_k) \end{bmatrix} \quad (5.6)$$

İlişki(M_i, M_k) ne kadar yüksekse, iki metot arasındaki ilişki o kadar yüksek değerdedir denilir ve sınıfın toplam ilişki değeri yükselir. Sınıfın toplam ilişki değeri Toplam_İlişki(X) ne kadar yüksekse sınıfın uyumu o kadar yükselir.

COMIAS'ın normalize bir ölçüt olması için bir sınıftaki maksimum ilişki değerini bulup bu sayıya yukarıdaki Toplam_İlişki(X) değerini oranlamak gerekmektedir. Bir sınıftaki maksimum ilişki değeri Max_İlişki(X), maksimum metot çağrımı ilişkisi ve maksimum üye alan paylaşımı ilişkisinin toplamı ile bulunur.

$$Max_İlişki(X) = Max(Metod_cag) + Max(Attr_pay) \quad (5.7)$$

Bir sınıftaki maksimum metot çağrımı ilişkisi şu şekilde tanımlansın:

Max (Metot_cag) = Her (M_i, M_k) ilişkisinde bir çağrım vardır demektir ve bu da mümkün olan tüm (M_i, M_k) ilişki sayısı ile bulunur:

$M_i \in M_x$ ve $M_k \in M_x$, $i \neq k$. (5.8)

X sınıfının m tane metodu varsa mümkün olan tüm (M_i, M_k) ilişki sayısı m.(m-1)/2 olacaktır. Her ilişkide bir çağrım olması durumu, maksimum metot çağrımı durumunu verecektir. Maksimum metot çağrımı şu şekilde ifade edilir:

$$Max(Metod_cag) = \frac{m.(m-1)}{2} \times 1 \qquad (5.9)$$

Tablo 5.4.1'de 4 adet metodu olan bir sınıf için bu durum örneklenerek açıklanmıştır. 4 adet metodu olan bir sınıf için tüm ilişkiler ve maksimum metot çağrımı ilişkisi bu tabloda verilmiştir; her metot çifti için aralarında bir çağrım vardır.

Aynı şekilde bir sınıftaki maksimum üye alan paylaşımı ilişkisi şu şekilde tanımlanır:

Max (Attr_pay) = Her (M_i, M_k) ilişkisinde sınıfta bulunan tüm üye alanlar paylaşılıyor demektir:

$M_i \in M_x$ ve $M_k \in M_x$, $i \neq k$. (6.10)

Tablo 5.4.1 Metotlar arasında maksimum çağrım ilişkisi örneği

	Metot çağrım durumu
(M_1, M_2)	1
(M_1, M_3)	1
(M_1, M_4)	1
(M_2, M_3)	1
(M_2, M_4)	1
(M_3, M_4)	1
Max (Metod_cag)	6 = 6 *1

Tablo 5.4.2 Metotlar arasında üye alan paylaşımı ilişkisi örneği

	Üye alan paylaşım durumu
(M_1, M_2)	4

(M_1, M_3)	4
(M_1, M_4)	4
(M_2, M_3)	4
(M_2, M_4)	4
(M_3, M_4)	4
Max(Attr_pay)	24 = 6 * 4

X sınıfının a tane üye alanı varsa, maksimum üye alan paylaşımı, üye alan sayısının mümkün olan tüm (M_i, M_k) ilişki sayısı m.(m-1)/2 ile çarpımı kadardır. Maksimum üye alan paylaşımı şu şekilde ifade edilir:

$$Max(Attr_pay) = \frac{m.(m-1)}{2} \times a \qquad (5.11)$$

Ölçütü normalize edebilmek için gereken Max_İlişki(X) değeri aşağıdaki gibi bulunur:

$$Max_İlişki(X) = \left(\frac{m.(m-1)}{2} \times 1\right) + \left(\frac{m.(m-1)}{2} \times a\right) \qquad (5.12)$$

Yukarıdaki tanımlamalardan yola çıkarak, önerilen ölçüt aşağıdaki gibi formüle edilir:

$$COMIAS = \frac{Total_İlişki(X)}{Max_İlişki(X)} \qquad (5.13)$$

COMIAS ölçütü [0, 1] arasında normalize edilmiştir. 0 düşük uyumu, 1 yüksek uyumu göstermektedir.

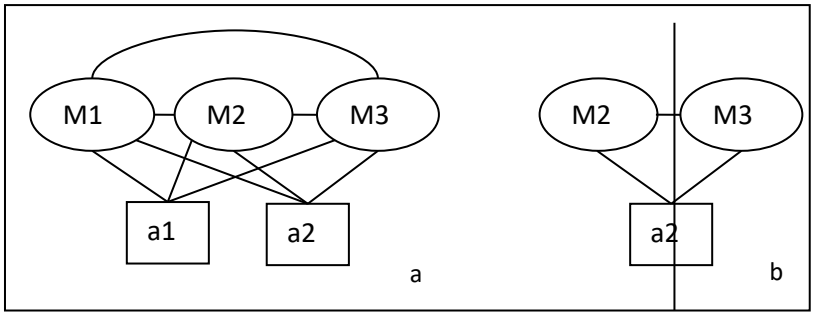
5.5. COMIAS Uyum Ölçütünün Örneklenmesi

Bu bölümde, verilen hipotezler doğrultusunda yapılan örnekler ile COMIAS ölçütünün verdiği sonuçlar değerlendirilecektir.

HİPOTEZ 1:

"Max. ve min. değerler sezgisel olarak aynı elde etmeyi beklediğimiz örneklerde aynı çıkmalıdır"

Maximum ilişki:



Şekil 5.5 Maximum ilişki grafiği örnekleri

Şekil 5.5 a ve b grafikleri maximum ilişki grafiklerine örnektir. Her iki grafik için ölçüt maximum uyum değeri olan "1" sonucunu vermelidir.

a grafiği için ölçüt hesaplaması:

m = 3, a = 2

Tablo 5.5.1 Maximum ilişki grafiği a için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	2
(M_1, M_3)	1	2
(M_2, M_3)	1	2

$$COMIAS = \frac{\left[(1+2)+(1+2)+(1+2)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 2\right)} = \frac{9}{9} = 1$$

b grafiği için ölçüt hesaplaması:

m = 2, a = 1

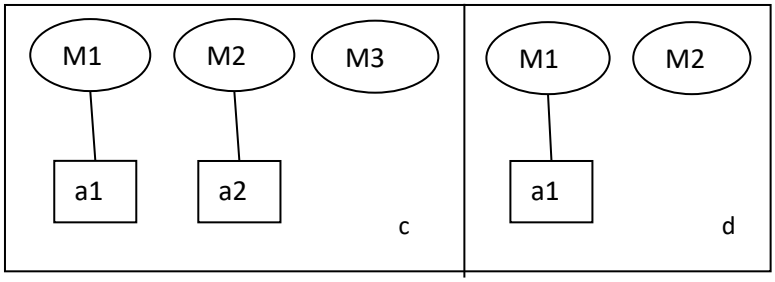
Tablo 5.5.2 Maximum ilişki grafiği b için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	1

$$COMIAS = \frac{[(1+1)]}{\left(\frac{2\times(2-1)}{2}\times 1\right)+\left(\frac{2\times(2-1)}{2}\times 1\right)} = \frac{2}{2} = 1$$

Her iki grafik için ölçüt maximum uyum değeri olan "1" sonucunu verdi.

Minimum ilişki:



Şekil 5.5.1 Minimum ilişki grafiği örnekleri

Şekil 5.5 c ve d grafikleri minimum ilişki grafiklerine örnektir. Her iki grafik için ölçüt minimum uyum değeri olan "0" sonucunu vermelidir.

c grafiği için ölçüt hesaplaması: m = 3, a = 2

Tablo 5.5.3 Minimum ilişki grafiği c için ölçüt metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	0	0
(M_1, M_3)	0	0
(M_2, M_3)	0	0

$$COMIAS = \frac{[(0+0)+(0+0)+(0+0)]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 2\right)} = \frac{0}{9} = 0$$

d grafiği için ölçüt hesaplaması: m = 2, a = 1

Tablo 5.5.4 Minimum ilişki grafiği d için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	0	0

$$COMIAS = \frac{[(0+0)]}{\left(\frac{2\times(2-1)}{2}\times 1\right)+\left(\frac{2\times(2-1)}{2}\times 1\right)} = \frac{0}{2} = 0$$

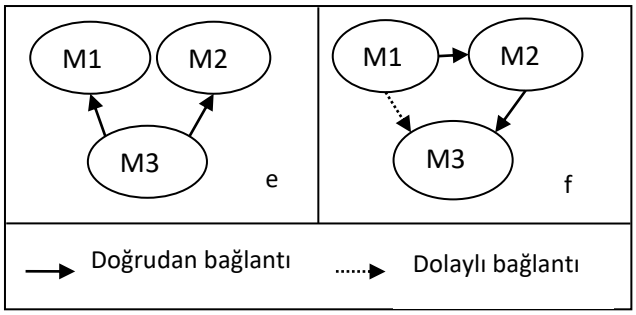
Her iki grafik için ölçüt minimum uyum değeri olan "0" sonucunu verdi.

Görülüyor ki, ölçüt sezgisel olarak uyum özelliği aynı beklenen sınıflar için aynı sonucu verdi. Aynı zamanda, sonuçlarda herhangi bir belirsizlik ya da tanımsız bir durum oluşmadı.

HİPOTEZ 2:

"Ölçütte dolaylı bağlantıyı hesaba katmazsak uyum özelliğini iyi yakalamamış oluruz. Bu durumda ölçüt daha düşük bir sonuç vermeli".

Dolaylı bağlantı hesaba katılmazsa:



Şekil 5.5.2 Dolaylı bağlantı değerlendirmesi için grafik örnekleri

Dolaylı bağlantı hesaba katılmazsa e grafiği için ölçüt hesaplaması:

Tablo 5.5.5. Dolaylı bağlantı hesaba katılmadan e grafiği için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu

(M_1, M_2)	0	0
(M_1, M_3)	1	0
(M_2, M_3)	1	0

$$COMIAS = \frac{\left[(0+0)+(1+0)+(1+0)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 0\right)} = \frac{2}{3}$$

Dolaylı bağlantı hesaba katılmazsa f grafiği için ölçüt hesaplaması:

Tablo 5.5.6 Dolaylı bağlantı hesaba katılmadan f grafiği için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	0
(M_1, M_3)	0	0
(M_2, M_3)	1	0

$$COMIAS = \frac{\left[(1+0)+(0+0)+(1+0)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 0\right)} = \frac{2}{3}$$

Dolaylı bağlantı hesaba katılmazsa e ve f grafikleri için ölçüt aynı sonucu veriyor. Ama iyi bir ölçüt farkı gözetebilmeli.

Dolaylı bağlantı hesaba katılırsa:

Dolaylı bağlantı hesaba katılırsa e grafiği için ölçüt hesaplaması değişmiyor çünkü dolaylı bağlantı yok.

Dolaylı bağlantı hesaba katılırsa f grafiği için ölçüt hesaplaması:

$$COMIAS = \frac{\left[(1+0)+(1+0)+(1+0)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 0\right)} = \frac{3}{3} = 1$$

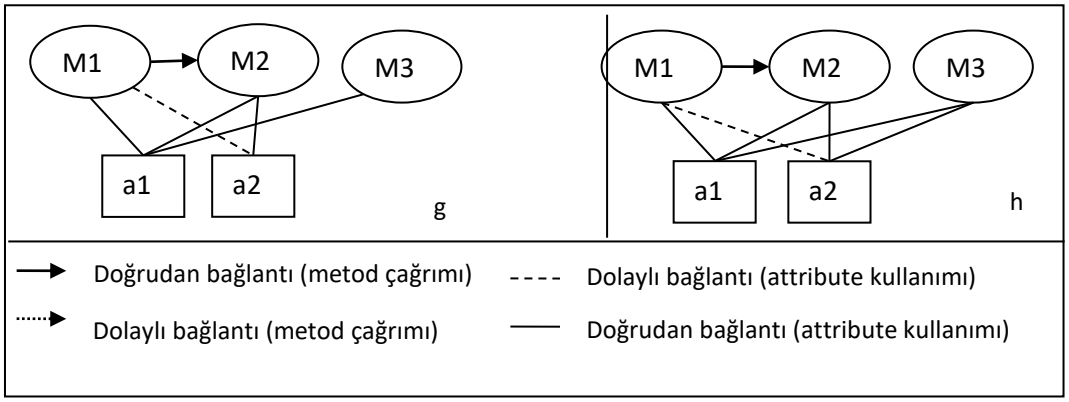
Dolaylı bağlantı hesaba katılırsa sezgilerle paralel sonuç alınıyor. f grafiğinin uyum değeri daha iyi çıktı. Dolaylı bağlantıları hesaba katmanın önemini bu örnekle görebiliyoruz. Dolaylı bağlantıları görebilmenin tek yolu da yönlü grafik çizmektir

Tablo 5.5.7 Dolaylı bağlantı hesaba katılırsa f grafiği için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	0
(M_1, M_3)	1	0
(M_2, M_3)	1	0

HİPOTEZ 3:

"Sınıf bileşenleri arasındaki etkileşim arttıkça uyum değeri daha iyi çıkmalı".



Şekil 5.5.3 Etkileşim değerlendirmesi için grafik örnekleri

Şeki 5.5.3 grafiğine sezgisel olarak bakıldığında h grafiğinin uyum değeri g grafiğinden daha iyi çıkmalı.

g grafiği için ölçüt hesaplaması: m = 3, a = 2

$$COMIAS = \frac{\left[(1+2)+(0+1)+(0+1)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 2\right)} = \frac{5}{9}$$

h grafiği için ölçüt hesaplaması: m = 3, a = 2

Tablo 5.5.8 g grafiği için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	2
(M_1, M_3)	0	1
(M_2, M_3)	0	1

Tablo 5.5.9 h grafiği için metot çağrım ve üye alan paylaşım durumu

	Metot çağrım durumu	Üye alan paylaşım durumu
(M_1, M_2)	1	2
(M_1, M_3)	0	2
(M_2, M_3)	0	2

$$COMIAS = \frac{\left[(1+2)+(0+2)+(0+2)\right]}{\left(\frac{3\times(3-1)}{2}\times 1\right)+\left(\frac{3\times(3-1)}{2}\times 2\right)} = \frac{7}{9}$$

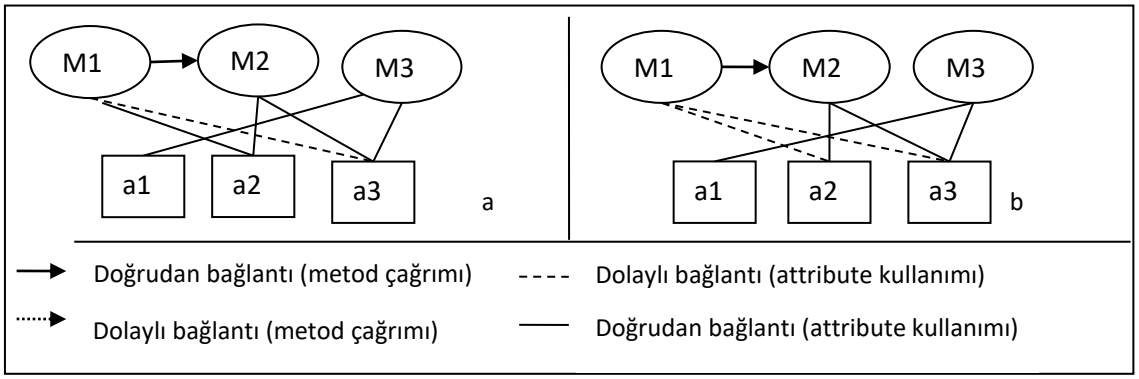
Sınıf bileşenleri arasındaki etkileşim arttıkça uyum değerinin daha iyi çıktığı görülmüştür.

Değerlendirme:

- Ölçüt tanımında, sınıfın en önemli elemanları olan üye alan ve metot varlıkları göz önüne alınmıştır.
- Önerilen ölçüt [0, 1] arasında normalize edilmiştir. “0” düşük uyumu, “1” yüksek uyumu gösterir.
- Sezgilerle paralellik gösteriyor. 2 tane farklı uyum özelliğine sahip sınıf farklı değer alıyor.
- Metot çağrımları göz önüne alınmıştır.
- Dolaylı bağlantılar göz önüne alınmıştır.
- Ölçüt, matematiksel olarak belirsiz ve tanımsız sonuçlar vermiyor.
- Özel metotlardan erişim metotları *uyumu* yapay olarak düşürürken constructor metotlar *uyumu*

yapay olarak arttırmaktadırlar. Literatürdeki ölçütlerin incelendiği 4. bölümde birçok ölçüt bu yönde sonuç vermiştir. COMIAS ölçütü

5.6. Önerilen Ölçüt ve LCOM Ölçütlerinin Bir Örnek Üzerinde İncelenmesi



Şekil 5.6.1 LCOM ölçütleri ve COMIAS değerlendirmesi için grafik örnekleri

5.6.1. LCOM1

a şıkkına göre LCOM1:

$\{I_1\} = \{a2\}, \{I_2\} = \{a2, a3\}, \{I_3\} = \{a1, a3\}$

LCOM1 = 1

b şıkkına göre LCOM1:

$\{I_1\} = \{-\}, \{I_2\} = \{a2, a3\}, \{I_3\} = \{a1, a3\}$

LCOM1 = 2

Değerlendirme:

- LCOM1 ölçütü 1 ve 2 değerleri vermiş, normalize bir durum görülmemektedir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak LCOM1 aynı değeri vermeli ancak farklı değer vermiş.
- Metot çağrımları göz önüne alınmamış.
- Dolaylı bağlantılar göz önüne alınmamış.

5.6.2. LCOM2

a şıkkına göre LCOM2:

P = 1, Q = 2

LCOM2 = P < Q = 0

b şıkkına göre LCOM2:

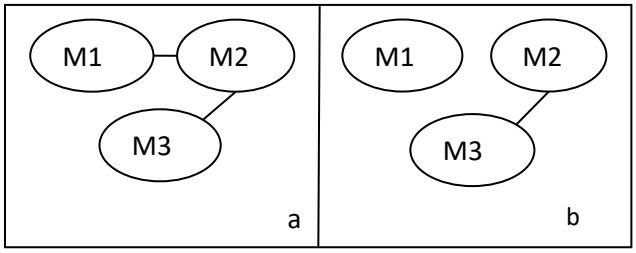
P = 2, Q = 1

LCOM2 = P – Q = 2 – 1 = 1

Değerlendirme:

- LCOM2 ölçütü 0 ve 1 değerleri vermiş, normalize bir durum görülmemektedir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak LCOM2 aynı değeri vermeli ancak farklı değer vermiş.
- Metot çağrımları göz önüne alınmamış.
- Dolaylı bağlantılar göz önüne alınmamış.

5.6.3. LCOM3



Şekil 5.6.2 LCOM3 ölçütü a ve b G_x grafikleri

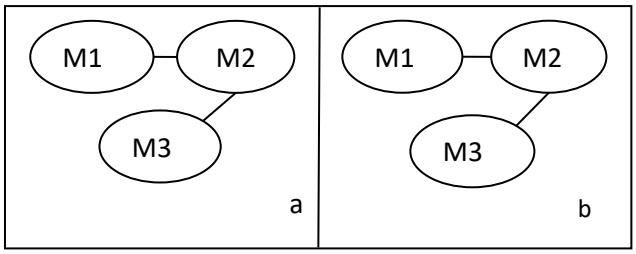
a şıkkına göre LCOM3 = 1

b şıkkına göre LCOM3 = 2

Değerlendirme:

- LCOM3 ölçütü 1 ve 2 değerleri vermiş, normalize bir durum görülmemektedir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak LCOM3 aynı değeri vermeli ancak farklı değer vermiş.
- Metot çağrımları göz önüne alınmamış.
- Dolaylı bağlantılar göz önüne alınmamış.

5.6.4. LCOM4



Şekil 5.6.3 LCOM4 ölçütü a ve b G_x grafikleri

a şıkkına göre LCOM4 = 1

b şıkkına göre LCOM4 = 1

Değerlendirme:

- LCOM4 ölçütü 1 değeri vermiş, normalize bir durum görülmemektedir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak LCOM4 aynı değeri vermeli. LCOM4 aynı değeri vermiş.
- Metot çağrımları göz önüne alınmış.
- Dolaylı bağlantılar göz önüne alınmamış.

5.6.5. LCOM5

a şıkkına göre LCOM5:

m = 3, a = 3

$$LCOM5 = \frac{\frac{1}{3}(1+2+2) - 3}{1-3} = 0.66$$

B şıkkına göre LCOM5:

m = 3, a = 3

$$LCOM5 = \frac{\frac{1}{3}\left(1+1+2\right) - 3}{1-3} = 0.83$$

Değerlendirme:

- LCOM5 ölçütü 0.66 ve 0.93 değerlerini vermiş. Örnekte sınıfın her bir üye alanı en az bir metot tarafından referans ediliyor, bu koşulda [0, 1] arasında kısmen normalize bir durum görülmektedir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak LCOM5 aynı değeri vermeli ancak farklı değer vermiş.
- Metot çağrımları göz önüne alınmamış.
- Dolaylı bağlantılar göz önüne alınmamış.

5.6.6. COMIAS

a şıkkına göre COMIAS:

$$COMIAS = \frac{\left[((1+2)+(0+1))+((0+1))\right]}{\left(\frac{3\times(3-1)}{2}\right)\times(3+1)} = \frac{5}{12} = 0.41$$

b şıkkına göre COMIAS:

$$COMIAS = \frac{\left[((1+2)+(0+1))+((0+1))\right]}{\left(\frac{3\times(3-1)}{2}\right)\times(3+1)} = \frac{5}{12} = 0.41$$

Değerlendirme:

- COMIAS ölçütü 0.41 değerini vermiş. Normalize bir durum görülmektedir. Ölçüm sonucu 0 ve 1 arasında bir değerdir.
- Metot çağrımları da dikkate alınıp bakıldığında sezgisel olarak COMIAS aynı değeri vermeli. COMIAS aynı değeri vermiş.
- Metot çağrımları göz önüne alınmış.
- Dolaylı bağlantılar göz önüne alınmış.

LCOM1, LCOM2, LCOM3 ve LCOM5 ölçütlerinin sonucuna göre b şıkkının uyum değeri daha düşük çıktı. Oysa a ve b şıklarının uyum değerleri aynı çıkmalı. Bu ölçütler metot çağrımlarını hesaba katmadıkları için uyum özelliğini iyi yakalayamayabiliyorlar.

Metot çağrımını hesaba kattığı için LCOM4 ölçütü a ve b şıkları için aynı sonucu verdi fakat LCOM4 ölçütünün normalizasyonu olmadığı için yorumlamak kolay değildir. Zaten LCOM4 ölçütü için bilinen bir gerçek vardır ki o da şu; LCOM4=1 değeri LCOM4 ölçütü için "genelde" elde edilen değerdir.

COMIAS ölçütünde ise sezgilerle paralel olarak a ve b şıkları için elde edilen değerler aynıdır. COMIAS ölçütü için normalizasyondan dolayı 0'ın düşük uyumu 1'in yüksek uyumu ifade ettiği bilindiğinden, çıkan 0.41 değeri a ve b şıklarında verilen örnek grafiklerin pek iyi bir uyum özelliğine sahip olmadıklarını gösteriyor.

5.7. Deneysel Geçerleme

5.7.1. Kullanılan Program

Chidamber ve Kemerer'in nesneye yönelik ölçütlerini hesaplayan çeşitli programlar bulunmaktadır. Bunların bir kısmı ücretli, bir kısmı ise ücretsizdir. Ücretli alternatiflerde, Essential Metrics, JStyle, JHawk ve Analyst4j gibi programlar göze çarpmaktadır. JMetric, Byte Code Metric Library (BCML) ve ckjm (Chidamber and Kemerer Java Metrics) ise ücretsiz olanlardır.

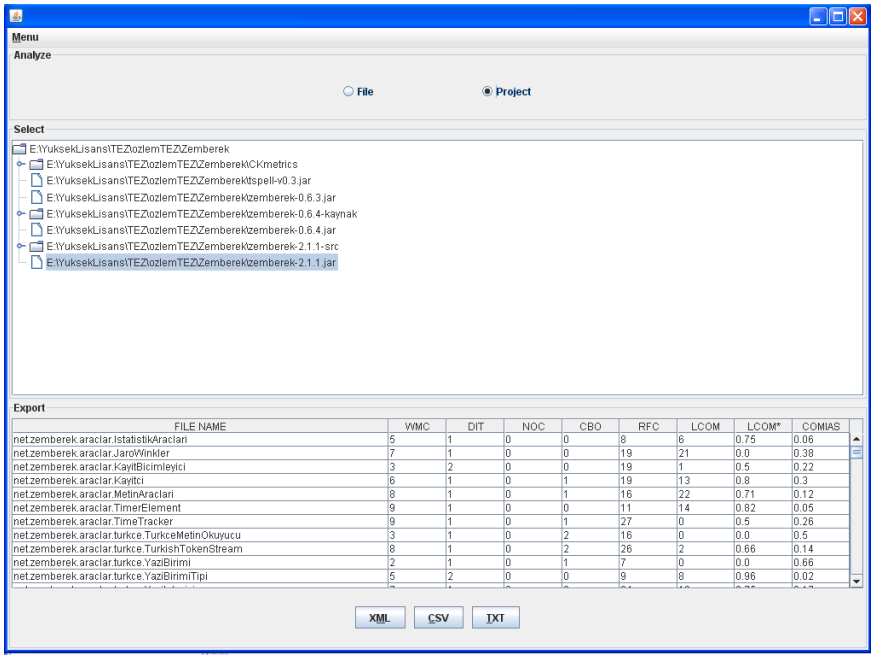
Deneysel geçerlemede kullanılacak yazılım daha önce "Nesneye Yönelik Programlamada Kalite Ölçütlerinin Hesaplanması" (Seri, 2008) adlı ara proje kapsamında açık kaynak, Java programlama dili ile geliştirilmiş ckjm tabanlı yazılım kalite ölçüm programıdır.

Ckjm, derlenmiş Java dosyalarının byte kodunu işlemden geçirerek Chidamber ve Kemerer nesneye dayalı ölçütlerini ölçer. Program her sınıf için Chidamber ve Kemerer tarafından teklif edilen altı ölçütü de hesaplar.

"Nesneye Yönelik Programlamada Kalite Ölçütlerinin Hesaplanması" (Seri, 2008) adlı bilgisayar projesi dersi kapsamında ckjm kodlarının, işlevini kullanıcı dostu bir şekilde yerine getirebilmesi için bir arayüz oluşturulmuştur. Arayüz geliştirmenin yanı sıra LCOM5 ölçütü kodlaması eklenerek hesaplanması sağlanmıştır.

COMIAS ölçütünün deneysel geçerlemesini yapabilmek amacıyla "Nesneye Yönelik Programlamada Kalite Ölçütlerinin Hesaplanması" (Seri, 2008) adlı ara proje çalışması kullanılmıştır. Ara proje çalışmasına COMIAS ölçütünün hesaplanması için gerekli olan kodlama eklentileri yapılmıştır. Mevcut çalışmada ölçülen ölçütler için var olan kodlama COMIAS ölçütünün hesaplanması için yeterli değildi. COMIAS ölçütü dolaylı bağlantıları ve metot çağrımlarını içerdiğinden ilave eklentiler yapma ihtiyacı doğdu. Gerekli olan tüm kodlama

eklentileri sağlanarak aynı yazılımdan COMIAS ölçütünün de hesaplanması sağlanmıştır. Oluşturulan programın arayüzünün gösterimi Şekil 13'de verilmektedir. Ölçütlerin hesaplama sonuçları arayüzden görülebilmektedir.



Şekil 5.7.1 Program arayüzü

5.7.2. Kalitesi Ölçülecek Yazılımın Seçimi

COMIAS uyum ölçütünün deneysel olarak geçerlenmesi amacıyla Zemberek adı verilen açık kaynak kodlu projenin çeşitli sürümleri kullanılmıştır. Zemberek, açık kaynak kodlu Türkçe doğal dil işleme kütüphanesi ve OpenOffice eklentisidir. Zemberek'in Java'da geliştirilmiş olması, kalite açısından aralarında karşılaştırma yapılabilecek farklı sürümlerinin bulunması ve nesne tabanlı olması, bu programın seçilmesinde etkili olmuştur.

5.7.3. Ölçüm Sonuçları

COMIAS tarafından üretilen sonuçların gerçek dünya verileriyle karşılaştırılabilmesi için kullanıcı görüşlerine başvurulmuştur. Bu amaçla Zemberek programı geliştirici grubundan olan Ahmet Afşın AKIN ile görüşülmüştür ve şu bilgiler elde edilmiştir:

- Tspell (Zemberek programının eski adı) 0.3 daha çok bir deneme sürümüdür. Sistem sadece belirli işlemleri yüksek sayılmayacak bir başarı ile yapabilmektedir.
- Zemberek 0.6.3 ve 0.6.4 arasında aslında çok fazla fark bulunmamaktadır. Çünkü asıl zıplama Zemberek 0.5 ile Zemberek 0.6 arasında gerçekleşmiştir.
- Zemberek 2.2.1 ile kod olgunluğa ulaşmıştır ve çalışmalar daha çok alt yapıda bazı düzenlemeler

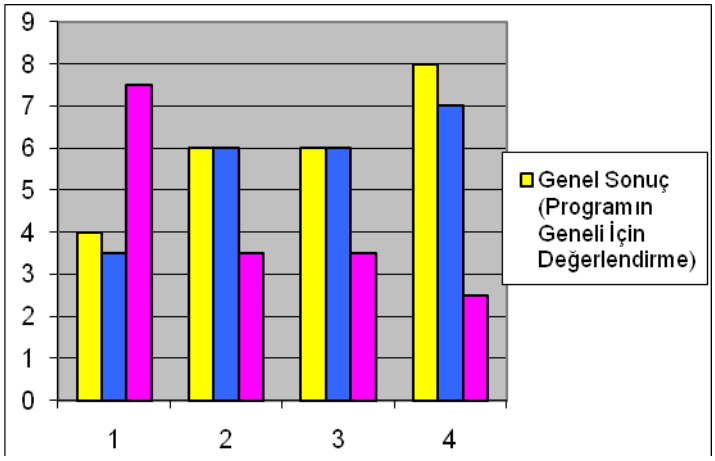
ile hata giderimi seklinde olmuştur.

Tablo 5.7.3 Zemberek programının farklı sürümleri için gerçek veriler

(1-2-3 ➔ Düşük / 4-5-6 ➔ Orta / 7-8 ➔ Yüksek / 9-10 ➔ Çok Yüksek**)**

Program Sürümleri	**Kod Kalitesi**	**Hata Oranı**	**Genel Sonuç (Programın Geneli İçin Değerlendirme)**
Tspell 0.3	3-4	7-8	4
Zemberek 0.6.3	6	3-4	6
Zemberek 0.6.4	6	3-4	6
Zemberek 2.2.1	7	2-3	8

Tablo 5.7.3'de Zemberek için kod kalitesi, hata oranı ve genel bir değerlendirme için ortalama sonuçlar gösterilmektedir. Tablo 6.13'den de anlaşılacağı üzere Zemberek 2.2.1 sürümünde kod kalitesi oldukça artmış, hata oranı ise azalmıştır.



Şekil 5.7.3. Sürümler için elde edilen gerçek verilerin grafiksel gösterimi

Bu bilgilere dayanarak, Zemberek programı için sürüm sayıları arttıkça genel olarak programın kalitesinin arttığı söylenebilir. COMIAS ölçütünün de bu görüşe paralel sonuçlar vermesi beklenmektedir.

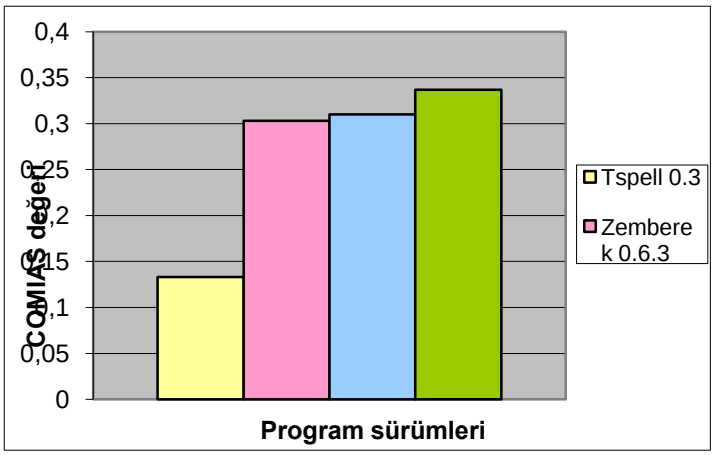
5.7.4. Zemberek Programı Ölçüm Sonuçları

Kalite ölçüm programında Zemberek programının farklı sürümleri kullanılarak elde edilen sonuçlar Tablo 5.7.4'te verilmiştir. Tablo 5.7.4'teki veriler programın her sürümü için ayrı ayrı hesaplanmıştır. Kalite ölçüm programında seçilen sürümün her sınıfı için elde edilen sonuçların aritmetik ortalaması alınarak elde edilen değer, sürümün seçilen ölçüt için genel değeri olarak alınmıştır.

Tablo 5.7.4 Zemberek programının ilerleyen sürümleri için COMIAS ölçütü ile elde edilen sonuçlar

Program Sürümleri:	Tspell 0.3	Zemberek 0.6.3	Zemberek 0.6.4	Zemberek 2.2.1
COMIAS ölçüm sonuçları:	0,133	0,303	0,310	0,337

Tablo 5.7.4 incelendiğinde COMIAS ölçütünün Zemberek ile ilgili gerçek verilerle paralel sonuçlar verdiği görülmektedir. COMIAS ölçütü ölçüm sonuçları sürümlerde uyumun ve kalitenin arttığını göstermektedir.



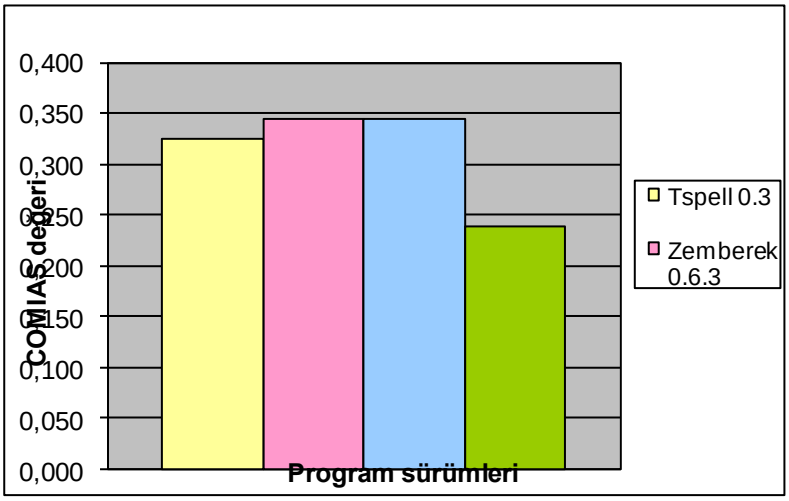
Şekil 5.7.4 Program sürümleri için elde edilen sonuçların grafiksel gösterimi

5.7.5. Paket Seviyesinde Ölçüm Sonuçları

Bu bölümde Zemberek yazılımının seçilen sürümlerinin hepsinde bulunan ortak paketler için ölçüm yapılmıştır. Sonuçlar Tablo 5.7.5'te görülebilir.

Tablo 5.7.5 Araçlar paketi için COMIAS ölçüm sonuçları

Program Sürümleri:	Tspell 0.3	Zemberek 0.6.3	Zemberek 0.6.4	Zemberek 2.2.1
COMIAS ölçüm sonuçları:	0,325	0,345	0,344	0,238



Şekil 5.7.5 Araçlar paketi için elde edilen sonuçların grafiksel gösterimi

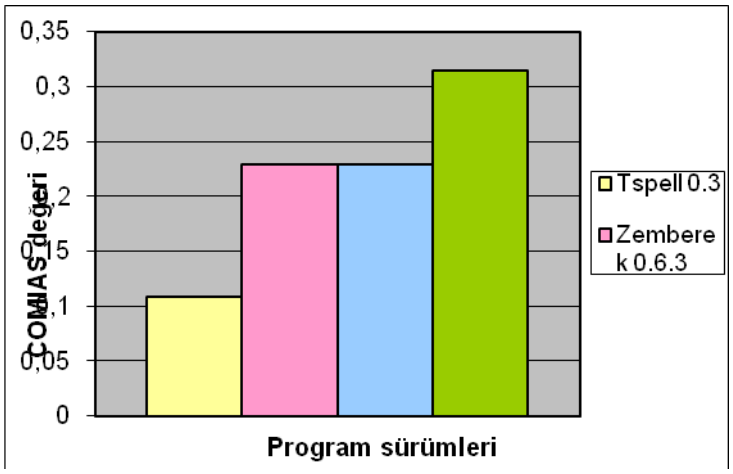
Araçlar paketi için elde edilen değerlere bakıldığında sürümler arttıkça kalitenin arttığı görülmemektedir (Tablo 6.15). Ancak ölçüm sonuçları pakette bulunan sınıf bazında detaylı incelendiğinde şu tespit yapılmaktadır:

- Tspell 0.3, Zemberek 0.6.3 ve Zemberek 0.6.4 sürümlerinde bulunan bir çok sınıf en yüksek uyum değerini (1) almış. Dolayısıyla da sürümlerin uyum değeri yapay olarak artmış. Gerçekte bu sınıflar metot sayısı 0 veya 1 olan sınıflardır. COMIAS ölçütü hesaplaması yapılırken metot sayısı 0 veya 1'e eşit olan sınıflara en yüksek uyum değeri atanmıştır. Buradan şu sonuca varılabilir; Eğer bir sürümde ya da sistemde metot sayısı 0 veya 1'e eşit sınıflar çok sayıda bulunuyorsa o zaman sürümün ya da sistemin uyumu yapay olarak artmaktadır. Elde edilen uyum değeri gerçek uyum değeri olmamaktadır. Bu bilgiler göz önüne alınarak COMIAS ölçütünün hesaplamasında her hangi bir yanlışlık veya eksiklik vardır diye düşünülmemelidir.

Yukarıda belirtilen tespit gereği Araçlar paketi için elde edilen uyum değerinin Tspell 0.3, Zemberek 0.6.3 ve Zemberek 0.6.4 sürümler için yapay artışlı değere sahip olduğu ve bu nedenle sürüm arttıkça kalitenin arttığı gözleminin bu paket için yapılamadığı görülmektedir.

Tablo 5.7.6 Yapı paketi için COMIAS ölçüm sonuçları

Program Sürümleri:	Tspell 0.3	Zemberek 0.6.3	Zemberek 0.6.4	Zemberek 2.2.1
COMIAS ölçüm sonuçları:	0,109	0,229	0,229	0,315

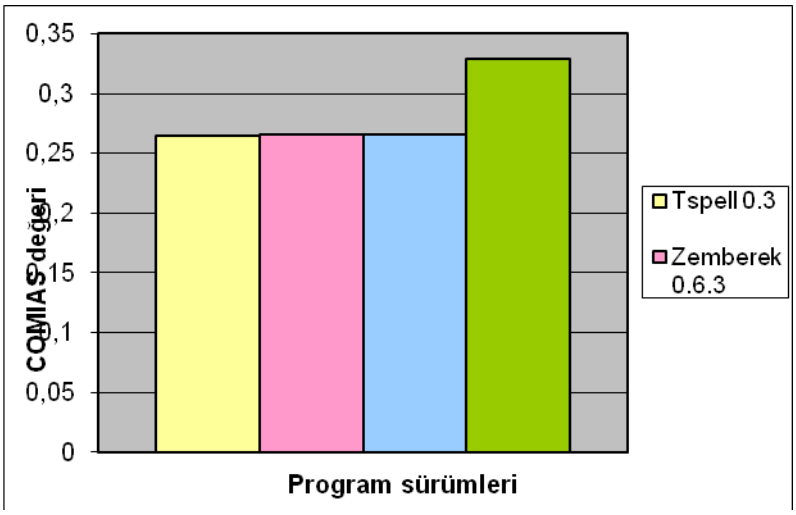


Şekil 5.7.6. Yapı paketi için elde edilen sonuçların grafiksel gösterimi

Yapı paketi için sürümlerde elde edilen değerlere bakıldığında sürümler arttıkça kalitenin arttığı net bir şekilde görülmektedir (Tablo 5.7.7). Sürümlerdeki Yapı paketi için elde dilen değerlerin, Zemberek 0.6.3 ve Zemberek 0.6.4 sürümlerinin gerçek verilerinde bu sürümlerin kalite açısından benzer olduğu bilgisini yansıttığı da görülmektedir.

Tablo 5.7.7 İşlemler paketi için COMIAS ölçüm sonuçları

Program Sürümleri:	Tspell 0.3	Zemberek 0.6.3	Zemberek 0.6.4	Zemberek 2.2.1
COMIAS ölçüm sonuçları:	0,265	0,266	0,266	0,329

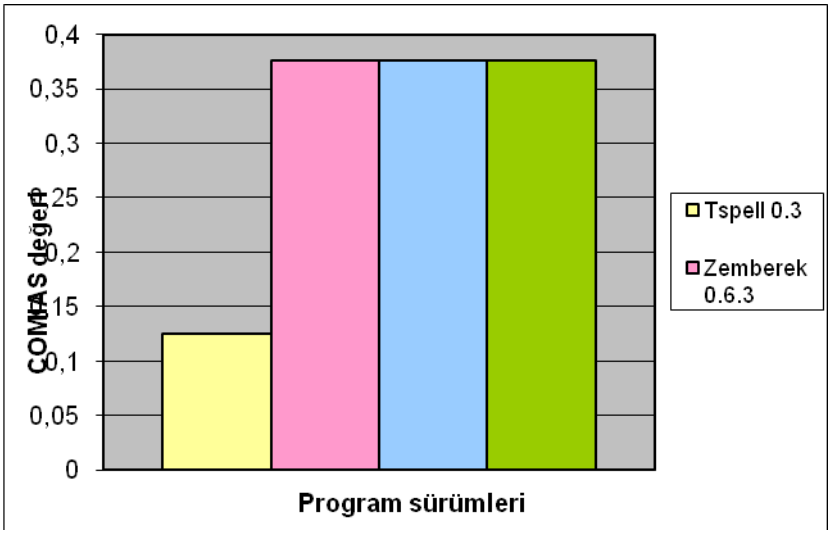


Şekil 5.7.7 İşlemler paketi için elde edilen sonuçların grafiksel gösterimi

Tspell 0.3 sürümünde bulunan İşlemler paketi için elde edilen değer Zemberek 0.6.3 ve Zemberek 0.6.4 sürümleri için elde edilen değere çok yakın çıkmışken Zemberek 2.2.1 sürümü için elde edilen değerin diğer sürümlerden farklılaştığı açıkça görülmektedir (Tablo 5.7.8). Bu sonuçlardan yola çıkılarak Zemberek 2.2.1 sürümünde işlemler paketi için genel olarak kalitenin arttığı net bir şekilde görülebilmektedir.

Tablo 6.18 İstatistik paketi için COMIAS ölçüm sonuçları

Program Sürümleri:	Tspell 0.3	Zemberek 0.6.3	Zemberek 0.6.4	Zemberek 2.2.1
COMIAS ölçüm sonuçları:	0,125	0,376	0,376	0,376



Şekil 6.19 İstatistik paketi için elde edilen sonuçların grafiksel gösterimi

Tspell 0.3 sürümünde bulunan İstatistik paketi için elde edilen değer diğer sürümlerden oldukça farklı çıkmıştır. Tspell 0.3 sürümünün ilk sürüm olduğu ve gerçek verilerle de kalite açısından zayıf olduğu bilindiğinden diğer sürümlerde kalitenin arttığı görülebilmektedir.

6. SONUÇLAR

Günümüz insanının çevresinde gelişen ve gittikçe artan yazılım sistemlerinde ortaya çıkabilecek bir yazılım hatası, insan hayatının yok oluşu, mali kayıp veya zaman gecikmesi gibi kötü sonuçlar doğurabilecek güce sahiptir. Bu nedenle, bugünün yazılım sistemleri hatasız ve tutarlı çalışabilen sistemler olarak çalışabilmek zorundadırlar. Artan yazılım kalitesi talebi, "kalite" karakteristiğini ürünler arasında önemli bir ayırıcı faktör olarak karşımıza çıkarmaktadır.

Yazılım uyumu, yazılımın içsel kalite özellikleri arasında bağlaşım ile birlikte önemli bir yere sahiptir ve bir modülün elemanlarının birbirine aitliğinin derecesi olarak tanımlanabilir. İyi tasarlanmış bir sistemde uyumun yüksek ve bağlaşımın düşük olması beklenir. Uyum eksikliği, sınıfların iki veya daha fazla alt sınıfa ayrılmaları gerekliliğini gösterir. Düşük uyum karmaşıklığı arttırarak geliştirme sürecindeki hataların olma olasılığını arttırır. Literatürde önerilen uyum ölçütlerinden hiçbiri bir standart olarak yaygın kabul görmemiştir (Poshyvanyk ve Marcus, 2005). Bu duruma yol açan etkenlerden biri de önerilen ölçütler ile ilgili daha fazla deneysel gerçekleme çalışmasına gereksinim duyulmasıdır (Briand vd., 1999).

Bu proje çalışmasında literatürde bulunan yapısal uyum ölçütleri incelenmiş, örneklerle birlikte değerlendirilmiştir.

Yapılan literatür araştırması ile elde edilen yapısal uyum ölçütlerinin incelenmesi sonucu ölçütlerin sınıflandırılmasının yararlı olacağı düşünülmüş ve bu kapsamda var olan bazı sınıflandırma kriterlerine ek olarak yeni kriterler tanımlanmıştır. Yapısal uyum ölçütlerinin her biri bu sınıflandırma kriterlerine göre değerlendirilmiştir.

Bu proje çalışmasında yapısal uyum ölçütlerinden yola çıkılarak temelde LCOM ölçütlerinin eksikliklerini adresleyen yeni bir ölçüt COMIAS önerilmektedir. COMIAS ölçütü ortaya konan görüşler ve hipotezlerle açıklanmaya çalışılmıştır. Yapılan örneklerle COMIAS, LCOM ölçütleri ile karşılaştırılmıştır. LCOM ölçütlerinin sezgilerle uyuşmayan sonuçlar verdiği durumlarda COMIAS ölçütünün doğru sonuçlar verdiği görülmüştür. Ölçütün Weyuker özellik kümesi kullanılarak teorik geçerlemesi yapılmıştır. Teorik geçerlemenin yanı sıra Zemberek adı verilen açık kaynak kodlu projenin çeşitli sürümleri kullanılarak COMIAS ölçütü deneysel olarak da geçerlenmiştir.

COMIAS ölçütü, normalize oluşu, dolaylı ilişkileri de göz önünde bulundurması ve sezgilerle paralel sonuçlar vermesi özellikleri ile ümit vericidir. Ancak COMIAS ölçütü kalıtım ile aktarılan metotları dikkate almamaktadır. Bu nedenlerle COMIAS ölçütünün deneysel geçerlemesi için çalışmaların sürdürülmesi ve kalıtım ilişkisinin de dikkate alındığı yeni bir uyum ölçütü üzerinde çalışılması planlanmaktadır.

7. REFERANSLAR

Abubakar, A., AlGhamdi, J. ve Ahmed M., "Can Cohesion Predict Fault Density", IEEE International Conference on Computer Systems and Applications, Vols 1-3.,2006.

Aman H., Yamasaki K., Yamada H. ve Noda M., "A Proposal of Class Cohesion Metrics Using Sizes of Cohesive Parts", Knowledge-based Software Engineering, T. welzer et al.(Eds.), pp102-107, IOS Press, Sept. 2002.

Badri, L. ve Badri, M. "A Proposal of a New Class Cohesion Criterion: An Empirical Study". Journal of Object Technology, 3(4), Apr. 2004. Special issue: TOOLS USA 2003.

Baldassari, B., Robach, C. ve du Bosquet, L., "Early Metrics for Object Oriented Designs", IEEE Testability Assessment, 2004. IWoTA 2004. Proceedings. First International Workshop p.62-69, 2004.

Bansiya, J. ve Davis, C. G., "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, vol. 28, no. 1, January 2002, pp. 4-17.

Bieman, J. ve Kang, B-K., "Cohesion and Reuse in An Object-Oriented System", in Proceedings of ACM Symposium on Software Reusability (SSR'95), April 1995, pp. 259-262.

Briand, L.C., Daly, J.C. ve Wüst J. "A unified Framework for Cohesion Measurement in Object-Oriented Systems", Technical Report ISERN-97-05. Kaiserlautern:Germany: Fraunhofer Institute for Experimental Software Engineering, 1997.

Briand, L.C., Daly, J.C. ve Wüst J. "A unified Framework for Cohesion Measurement in Object-Oriented Systems", Empirical Software Eng.; An Int'l J., vol.3, no. 1, pp. 65-117, 1998a.

Briand, L. C., Daly, J. W., Porter, V., ve Wüst, J., "A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems", in Proc. of International Software Metrics Symposium, Bethesda, MD, Nov. 21 1998b, pp. 43-53.

Briand, L. C., Daly, J. W., ve Wüst, J. K., "A Unified Framework for Coupling Measurement in Object-Oriented Systems." IEEE Transactions on Software Engineering, vol. 25, no. 1, January/February 1999.

Briand L., Morrasca S., and Basili V., "Defining and Validating Measures for Object-Based High-Level Design" IEEE Transactions On Software Engineering, vol. 25, no. 5, September/October 1999.

Briand, L. C., Wüst, J., Daly, J. W., ve Porter, V. D., "Exploring the Relationship between Design Measures and Software Quality in Object-Oriented Systems", Journal of Systems and Software, vol. 51, no. 3, May 2000, pp. 245-273.

Bunge, M., Treatise on Basic Philosophy: Ontology I : The Furniture of the World. Boston: Riedel, 1977.

Chidamber, S. R. ve Kemerer, C. F., "Towards a Metrics Suite for Object Oriented Design", in Proceedings of OOPSLA'91, 1991, pp. 197-211.

Chidamber, S. R. ve Kemerer, C. F., "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, vol. 20, no. 6, 1994, pp. 476-493.

Cho, E. S., Kim, C. J., Kim, D. D., ve Rhew, S. Y., "Static and Dynamic Metrics for Effective Object Clustering", in Proceedings of Asia Pacific International Conference on Software Engineering, 1998, pp. 78 - 85.

Counsell, S., ve Swift, S., "The Interpretation and Utility of Threee Cohesion Metrics for Object-Oriented Design", ACM Transactions on Software Engineering and Methodology, Vol. 15, No. 2, April 2006, Pages 123–149.

Ebert, C. ve Dumke, R., "Software Measurement: Establish - Extract - Evaluate – Execute", Springer, 2007.

El-Emam, K., "Object-Oriented Metrics: A Review of Theory and Practice", in Advances in software engineering, Springer-Verlag, New York, 2002, pp. 23-50.

Etzkorn, L. H. ve Davis, C. G., "Automatically Identifying Reusable OO Legacy Code", IEEE Computer, vol. 30, no. 10, October 1997, pp. 66-72.

Ertemel, H. Ö., Selçuk, Y. E. ve Kalıpsız, O., 'A Cohesion Metric Proposal for Object- Oriented Systems: COMIAS', 13th WSEAS Int'l. Conf. on Computers, Rhodes, Greece, 575-580, Temmuz, 2009.

Ertemel, H. Ö., Selçuk, Y. E. ve Kalıpsız, O., 'Nesneye Yönelik Sistemler İçin Bir Uyum Ölçütü Önerisi: COMIAS', IV. Ulusal Yazılım Müh. Semp., İstanbul, 161-166, Ekim, 2009.

Fowler, M., Beck, K., Brant, J. and Opdyke, W.: Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999

Gui, G. ve Scott, P.D., "Coupling and Cohesion Measures for Evaluation of Component Reusability", MSR'06, May 22-23, 2006, Shanghai, China.

Hitz, M. ve Montazeri, B., "Measuring Coupling and Cohesion in Object-Oriented Systems", Proceedings of International Symposium on Applied Corporate Computing. (Monterrey, Mexico, 1995).

Lee, J. K., Jung, S. J., Kim, S. D., Jang, W. H., ve Ham, D. H., "Component Identification Method with Coupling and Cohesion", in Proc. of 8th Asia-Pacific Software Engineering Conference (APSEC'01), Dec. 2001, pp. 79-86.

Lieberherr, K., Holland, I., ve Riel, A., "Object Oriented Programming: An Objective Sense of Style,'' in Third Annu. ACM Conf: Object Oriented Prog. Syst., Lung. and Applicat. (OOPSLA), 1988, pp. 323-334.

Maletic, J. I. ve Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", in Proceedings of 23rd International Conference on Software Engineering, Toronto, Canada, May 12-19 2001, pp. 103-112.

Meyers, T. M. ve Binkley, D., "Slice-based Cohesion Metrics and Software Intervention", in Proceedings of 11th Working Conference on Reverse Engineering (WCRE'04), Nov. 8-12 2004, pp. 256-265.

Seri, N., Bilgisayar Projesi Ders Raporu, "Nesneye Yönelik Programlamada Kalite Ölçütlerinin Hesaplanması", YTÜ, İstanbul 2008.

Quah, T.-S. ve Thwin, M. M. T., "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics", in Proceedings of International Conference on Software Maintenance, September 22-26 2003, pp. 116-125.

Poshyvanyk, D. ve Marcus, A., “The Conceptual COhesion of Classes”. 21st IEEE International Conference on Software Maintenance (ICSM'05)1063-6773/05 $20.00 © 2005 IEEE.

Stevens, W., Myers, G., ve Constantine, L., “Structured Design,” IBM Systems J., vol. 13, no. 2, pp. 115-139, 1974.

Zhou, Y., Xu, B., Zhao, J., and Yang, H., "ICBMC: An Improved Cohesion Measure for Classes", in Proceedings of International Conference on Software Maintenance, October 3-6 2002, pp. 44-53.

Proje Kapsamında Yapılan Yayınlar, Tezler

YAZAR (LAR)	MAKALE/BİLDİRİ BAŞLIĞI	DERGİ/ TOPLANTI ADI	CİLT/ SAYI/ SAYFA NO	TARİH	TEZ		TÜRÜ		BİLDİRİ		MAKALE*		
					YL	D	Y.İ	Y.D	PS	S	A	B	C
Hüsniye Özlem Ertemel	Nesneye Yönelik Yazılım Geliştirmede Kalite Ölçütlerinin İncelenmesi			2009	X								
H. Özlem Ertemel, Yunus Emre Selçuk, Oya Kalıpsız	A Cohesion Metric Proposal for Object-Oriented Systems: COMIAS	13th WSEAS Int'l. Conf. on Computers	575-580	2009				X		X			
H. Özlem Ertemel, Yunus Emre Selçuk, Oya Kalıpsız	Nesneye Yönelik Sistemler İçin Bir Uyum Ölçütü Önerisi: COMIAS	IV. Ulusal Yazılım Müh. Semp.	161-166	2009			X			X			
Nil Seri (Danışman: Yunus Emre Selçuk)	Nesneye Yönelik Programlamada Kalite Ölçütlerinin Hesaplanması	Bilgisayar Projesi Dersi (Lisans), Proje Raporu		2008									
Melek Çoruh (Danışman: Yunus Emre Selçuk)	Yazılım Kalite Değerlendirmesi	Bilgisayar Projesi Dersi (Lisans), Proje Raporu		2008									

YL: Yüksek Lisans Tezi D: Doktora Tezi Y.İ: Yurtiçi Y.D: Yurtdışı PS: Poster S: Sunum

* Uluslararası indekslere girmiş dergilerde yayınlanan makalelerin sınıflarına http://www.ulakbim.gov.tr/cabim/ubyt/dergiler.php adresinden ulaşılabilir.