Görselleştirmenin Yazılım Tasarımındaki Etkisi

Beyzanur DEMİR

Bilgisayar Mühendisliği Bölümü, Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi, Sakarya
beyzanur.demir@ogr.sakarya.edu.tr

Özet. "Bir resim bin sözcüğe bedeldir." denir. Bilgi görselleştirme, bir konuyu kavramayı kolaylaştırabilmek için etkileşimli görsel sunumların kullanımıdır. Yazılımdan elde edilen metrikleri incelerken, bu metrik değerleri görselleştirerek analiz etmek, görselleştirme tekniklerinin yazılımdaki uygulamalarının bir alt bölümü olarak karşımıza çıkmaktadır. Bu çalışmada literatürdeki görselleştirme tekniklerini yazılıma uygulayan çalışmalardan örnekler verilmiş ve sonrasında bu çalışmaların analizi yapılmıştır. Yapılan analiz sonucunda literatürdeki çalışmalardaki eksiklikler belirtilmiş ve bu eksikliğin giderilebilmesi için daha önce yapılmış bir çalışmayı geliştirerek, hangi yazılım için hangi görsel modelin kullanılabileceğine dair karar-destek sistemi içeren bir görselleştirme sistemi önerilmiştir. Literatürde bulunan görselleştirme tekniklerine değinilmiş ve ardından apartman dairesi metaforu kullanarak Java yazılımı için bir model oluşturulmuştur. Yapılan çalışmanın, daha önce yapılan çalışmalarla benzerlik, paralellik ve farklılıkları ile tartışılmış ve son kısımda da bilimsel/günlük hayata katkısı, kullanılabilirliği, literatüre ne kazandırdığı, teori ve uygulama açısından hangi kanılara varıldığı, kullanılabilirliğinden bahsedilmiştir.

Anahtar Kelimeler: Görselleştirme, metafor, yazılım modeli, görselleştirme teknikleri

The Effect of Visualization in Software Design

Abstract. "A picture is worth a thousand words." It called. Information visualization is the use of interactive visual presentations to facilitate understanding a subject. While examining the metrics obtained from the software, it is seen as a sub-section of the applications of visualization techniques in the software to visualize and analyze these metric values. In this study, examples of the studies that apply the visualization techniques in the literature are given and then these analyzes are analyzed. As a result of the analysis made, the deficiencies in the studies in the literature were identified and a visualization system including a decision-support system regarding which software model could be used for which software was used was developed by developing a previous study in order to overcome this deficiency. Visualization techniques in the literature are mentioned and then a model is created for Java software using the metaphor of the apartment. The similarity, parallelism and differences of the work done with the previous studies have been discussed and in the last part, it has been mentioned about its contribution to scientific / daily life, its usability, what it has brought to the literature, its opinions in terms of theory and practice.

Keywords: Visualization, metaphor, software model, visualization techniques

1.Giriş

1.1 Görselleştirmenin insan için önemi

İnsanlar uzun zamandır, sorunları anlamak ve çözüm üretebilmek için görselleştirmelerden faydalanmaktadır. Bilgisayarların grafiksel yetenekleri arttıkça, bilgisayarlar görselleştirmenin önemli bir aracı haline gelmiştir. "Bir resim bin sözcüğe bedeldir." denir ve bugün büyük veri çağında işletmeler çeşitli veri türlerinden, şirket içi ve bulut tabanlı kaynaklardan akan bilgi seline kapılmışken bu eski deyiş, hiç olmadığı kadar anlamlı hale geldi. Görseller analizi daha kolay ve daha anlaşılır hale getirirken önemli konuları da bir bakışta görme becerisi sunar. Dahası, çoğu insan görsellere metinden çok daha iyi tepki verir. Beyne gönderilen bilgilerin %90'ı görseldir ve beyin görselleri metne kıyasla 60.000 kat daha hızlı işler.[3] Bu noktalar, bilgiyi analiz etmek ve iletmek için veri görselleştirmesi kullanmanın önemini güçlü bir biçimde ortaya koymaktadır.

1.2 Görsel Zeka Nedir?

Görsel-Uzamsal zeka olarak da bilinen bu zeka türü, Howard Gardner'ın tanımladığı "Çoklu Zeka Kuramı" içerisinde yer alan zeka türlerinden biridir. Aynı zamanda fotoğrafik hafıza olarak da tanımlanır. Görme ile öğrenme, gördüklerini hızlı ve kolay biçimde akılda tutma ve analiz edip sonuca varma becerilerinin bütününe görsel zeka adı verilir. Geniş bir çerçevede düşünüldüğünde, görsel zeka için beynin ilk dili denilebilir. Beynimiz, dünyaya merhaba dediği andan başlayarak görüntüler ya da imajlar(resimler) ile düşünmeye başlar. Görsel zeka, adı üstünde; görme eylemi ile başlar. Hayal etmek ve görmek; gördüğümüz ya da hayal ettiğimiz şekiller, desenler, grafikler, soyut ve somut görüntüler, renkler ya da dokular görsel zeka ile ilişkilendirilir. Görsel-Uzamsal zeka, aynı zamanda nesnelerin, diğer nesnelerle ilişkisini de belirleyen etkenlerden biridir. Yer ve yön bulma becerilerimiz, mekansal farkındalık becerilerimiz ve içinde bulunulan ortamın farkında olmayı yöneten becerilerimizin gelişimi görsel-uzamsal zeka ile doğrudan bağlantılıdır.

1.3 Görsel Analitik Neden Önemlidir?

İyi veri görselleştirme verileri analiz etmek ve bu verilere dayalı kararlar almak için esastır. İnsanların yapılar ile ilişkileri hızla ve kolayca görüp anlamalarını, yalnızca ham sayılardan oluşan tablolarda fark edilemeyen trendleri tespit etmelerini sağlar ve birçok durumda, grafiklerde sunulanları yorumlamak için özel eğitim gerekmez, evrensel anlayış sağlanır.

İyi tasarlanmış bir grafik bilgi sağlamakla kalmaz, güçlü bir sunumla bu bilgilerin etkisini artırır, dikkat çeker ve insanların ilgisini hiçbir elektronik tablonun yapamayacağı ölçüde güçlü tutar.

1.4 Görselleştirme Teknikleri

Bir bilgi görselleştirme yöntemi ya da tekniği; özenle hazırlanmış bir kavrayışı, iç görüler elde edilmesini ve deneyimlerle iletişime geçilmesini sağlayan bilgiyi tasvir eden sistematik, kural tabanlı, dışsal, kalıcı ve grafiksel bir sunumdur [15]. Görselleştirme sürecinde mutlaka bir görselleştirme tekniği kullanılma zorunluluğu yoktur. Fakat veri kümesi karmaşıklaştıkça bir tekniğin kullanılması ihtiyacı da artmaktadır. Literatürde tanımlanmış her bir görselleştirme tekniğinin uygun olduğu en az bir veri kümesi bulunmaktadır. Literatürde bugüne kadar tanımlanmış 100'den fazla görselleştirme tekniği bulunmaktadır[15]. Literatürdeki çoğu görselleştirme değerlendirme yöntemi, kullanılabilirliğin kullanıcıların arayüz ile etkileşimi gözlemlenerek ölçüldüğü kullanıcı testlerine dayanmaktadır. Diğer kullanılabilirlik değerlendirme yöntemleri ise, sezgisel değerlendirme yöntemleridir [16, 17, 18]. Bu yöntemler, bir sistemin kullanılabilirlik uzmanları tarafından incelenerek, kullanılabilirlik problemlerinin ortaya çıkarılabildiği yöntemlerdir. Bilgi görselleştirme teknikleri kullanılabilirlik açısından değerlendirilirken, görsel sunum ve etkileşim mekanizmaları üzerinde incelenmesi gereken ölçütler farklı olabildiğinden ölçütler görsel sunum ve etkileşim olarak iki gruba ayrılmıştır. Buna ek olarak bazı ölçütler, hem görsel sunumu hem de etkileşimi ilgilendirmektedir. Bunlar görsel sunum ve etkileşim grubunda yer almıştır.

1.5 Yazılımda Görsellik

Bu çalışmada ele alınan konu ise görselleştirmenin yazılım tasarımı tarafındaki etkisi olacaktır. Yazılımın bakım, iyileştirme ve tekrar kullanma süreçlerinden sorumlu olan mühendisler öncelikli olarak üzerinde çalıştıkları yazılımı kavramak durumundadırlar. Endüstriyel kullanıma açık büyük ölçeklerdeki yazılımlarda böyle bir kavrayışa ulaşabilmek, özellikle belgelemenin yeteri kadar iyi olmadığı durumlarda, yoğun bir uğraş ve uzun zamanlar gerektirecektir. Yapılan araştırmalar bütün kaynakların %50 ile %75'lik kısmının yazılımın bakımı için ayrıldığını [1,2] ayrılan bu kaynağın büyük bir kısmının ise yazılımın kavranabilmesi için harcadığını göstermiştir. Tersine mühendislik sayesinde yazılımlardan elde edilen yazılım metrikleri yazılımların anlaşılabilmesi, kontrol edilmesi ve yazılım kalitesinin iyileştirilmesine olanak sunmaktadır [4,5]. Ancak çok büyük ölçekli yazılımlar için elde edilen yazılım metriklerinden yazılım kalitesi ile ilgili çıkarımlar yapmak zaman alacaktır. Ayrıca birden fazla metriği aynı anda değerlendirmek gerektiğinde tablolar halinde sunulan metriklerden gerekli bilgileri elde etmek kolay olmayacaktır [6]. Tam bu noktada

yazılım tasarımında görselleştirmenin, süreçleri kolaylaştırmadaki büyük rolünden bahsedilebilmektedir. Tasarım aşamasındaki görselleştirme insan beyninin karmaşıklığın birçok farklı yönünü paralel olarak işleyebilmesine olanak tanır.[7]

Bu çalışmada görselleştirme teknikleri kullanılarak, yazılım metriklerinden yazılım kalitesi hakkında fikir verebilecek bir görselliğe nasıl ulaşıldığı anlatılmıştır. 2.bölüm yazılım tasarımı için kullanılan görselleştirme çalışmalardan örnekler içermekte ve bu örneklerin analizi yapılmaktadır. 3.bölüm geliştirilen görselleştirme yönteminin tanıtımına ayrılmıştır. 4. bölümde Java yazılımı için bir metafor kullanılarak bir yazılım modeli tasarlanmıştır. 5.bölümde Yapılan çalışmanın, daha önce yapılan çalışmalarla benzerlik, paralellik ve farklılıkları ile tartışıldığı kısımdır ve son bölümde yapılan çalışmanın bilimsel/günlük hayata katkısı, literatüre ne kazandırdığı, teori ve uygulama açısından hangi kanılara varıldığının yazıldığı kısımdır.

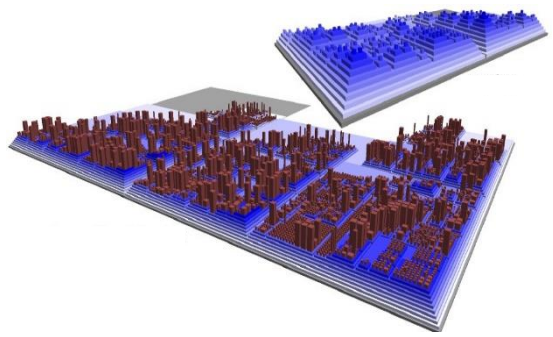
2.Literatürde Bulunan Yazılım Tasarımı için Kullanılan Görselleştirme Çalışmaları

Daha önceden de yapılan bu çalışmalarda kullanılan çeşitli metaforlar ile yazılımın çözüm sunduğun problemin anlaşılabilirliğini arttırmak ve bu görsel tasarım ile yazılımcıya yazılımı anlama ve geliştirebilme kolaylığı sağlamak hedeflenmiştir.

Bu bölümde konu hakkındaki benzer çalışmalardan örnekler verilmiştir.

➢ *Şehir Modeli*

Bu çalışmada[8] şehir metaforu kullanılarak gerçeklik hissi yüksek bir görselleştirme sağlanmıştır.



Şekil 1

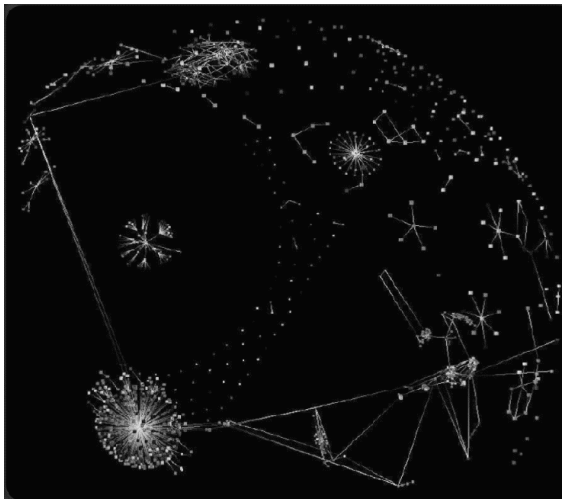
Yazılımdaki her sınıf kahverengi bir çubukla temsil edilmiştir. Çubuğun yüksekliği metot sayısı (NOM) ile çubuğun eni ve boyu ise sınıfın nitelik sayısı (NAO) ile orantılı yapılmıştır. Aynı paket içindeki sınıflar bir bölgede toplanarak şehrin mahallerini oluşturması sağlanmıştır.

Kullanılabilen metrik sayısının sınırlı olduğu bu çalışmada görselliğin oluşturulması sınıf sayısı arttıkça zorlaşmaktadır ve bu durum çok büyük ölçekli yazılımların görselleştirmesi sırasında zorluk çıkartabilmektedir.

Şehir metaforu kullanılarak oluşturulmuş bu çalışmaya benzer bir başka çalışmada[9] ise programın akışının şehrin içinden akan araçlar ile temsil edilmesi düşünülmüş fakat çalışma fikir bazında kalmıştır.

➢ *Güneş Sistemi Modeli*

Bu çalışmada[9] ise birçok alanda tanıdık bir metafor olarak karşımıza çıkan güneş sistemi kullanılmıştır.



Şekil 2

Evrenin merkezindeki gezegenler kalıtım ağacının tepesindeki sınıflar ile uydular ise bu sınıflardan türetilmiş olan sınıflar ile temsil edilmiştir. Gezegenlerin çapları da sahip olunan metot sayısı ile orantılı tutulmuştur. Az sayıda sınıfa sahip yazılımları görselleştiren bu çalışmanın, endüstriyel bir yazılım için uygulaması bulunmamaktadır.

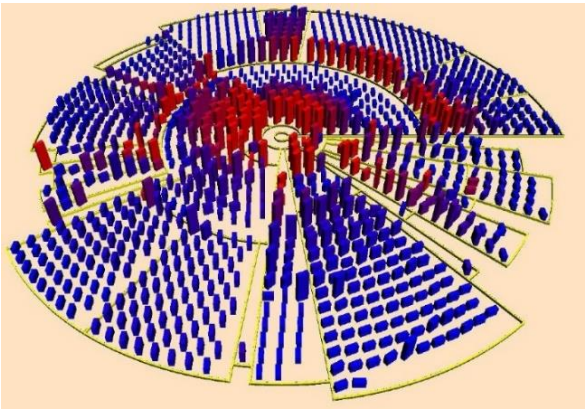
Benzer bir başka çalışmada[10] ise yazılım metrikleri evren benzeri bir model ile temsil edilmiştir. Boyut indirgeme algoritmaları kullanılarak metrikler üç boyuta indirilmiştir. 3 boyutlu bir görsellik sunan bu yöntemde, kullanıcının evren içinde kaybolma riski vardır.

Şekil 2'deki görselde sınıfların kalıtım hiyerarşisi görülmektedir. VRML aracının temel gezinme teknikleriyle kullanıcının model içinde gezinme olanağı vardır ancak daha derin ölçeklerde kullanıcının görsellik içinde kaybolma durumu da oluşabilmektedir.

➢ *Doku ve Renk Kullanımının Ağırlıklı Olduğu Modeller*

[11]'deki bu çalışma iki boyutlu bir görsellik sunarken, insanın görsel algılama yetisinin karmaşık görsellikleri de rahatça kavrayabileceği varsayarak doku ve rengi birlikte kullanmıştır.

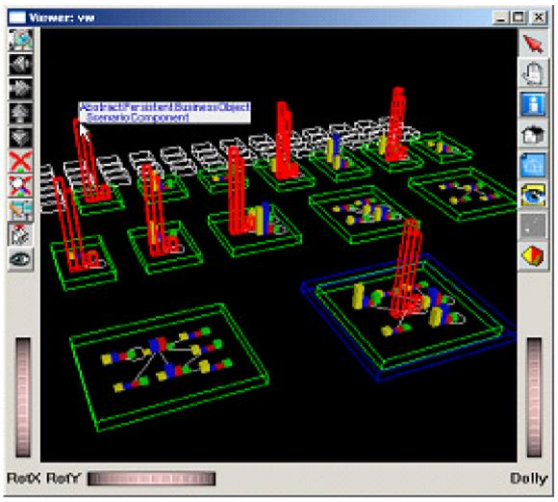
Sınıfın dışa bağımlığı (Fan-Out) arttıkça daha karmaşık dokular, sınıfa olan bağımlılık (Fan-In) arttıkça ise daha koyu renkler kullanılmıştır. Bu iki metriğin aynı anda yüksek olması yazılım tasarımında istenilen bir durum değildir. Çalışmada potansiyel olarak problemli olan sınıfların hızlıca fark edilmesini sağlayan bir görselleştirme önerilmiştir. Yazılım boyutu arttıkça her bir sınıfa karşı düşen dokunun hesaplanması zaman alacağı ve sınıf sayısı arttıkça görsellikte her bir sınıfa ayrılan alan azalacağı için büyük ölçekli yazılımlar için uygun bir çalışma değildir.



Şekil 3

Şekil3'teki görselde bir örneği gösterilen çalışmada 3 boyutlu kutular görselleştirme yapıtaşları olarak kullanılmıştır[12]. Kutuların en, boy ve yükseklikleriyle birlikte dizilişlerinde yaptıkları açı ve renkleri de ayrı birer metriği temsil etmek üzere kullanılmıştır. Sınıf sayısı arttıkça görselleştirmenin yapılması ve bu üç boyutlu görsellik içerisinde gezinilmesi zorlaşmaktadır.

➢ *UML Diyagramları için Oluşturulan Model*

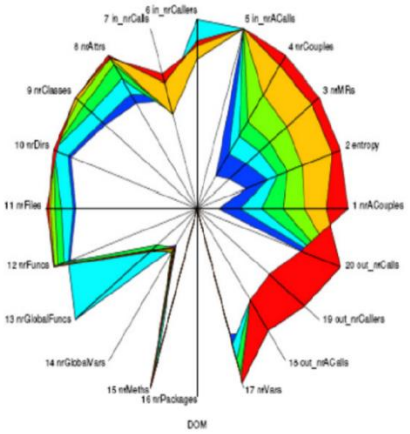


Şekil 5

Bu çalışmada[13] UML diyagramlarındaki her sınıfa karşı düşen dikdörtgenler üzerine sınıfa ait metrikleri gösteren üç boyutlu bir kutu eklenmiştir. Yazılım boyutunun büyümesi UML diyagramlarının anlaşılabilirliğini azalttığı için bu görselliğin de algılanması zorlaşacaktır.

➢ *Yazılım Evreni Modeli*

[14]'deki bu çalışma yazılımın farklı sürümlerinden elde ettiği 20 farklı metriği yazılımın 7 farklı sürümü için Kiviat diyagramı kullanarak bir arada vermiştir.



Şekil 4

Şekil 5'te yazılımın farklı sürümleri için metriklerinin aldığı değerler tek bir görsellikle özetlemiştir. Bu çalışmada sürümler arttıkça azalan metrik değerlerinin ara sürümlerde aldığı değerleri görmek mümkün olamamaktadır.

2.1 Literatürdeki Yazılım Tasarımı için Kullanılan Görselleştirme Çalışmalarının Analizi

Örnekleri verilen çalışmaların her birinin dayandığı bir görselleştirme esası bulunmaktadır. Örneklerde de görülebileceği üzere belli başlı metaforlar kullanılarak bu esaslar gerçekleştirilmiştir. Bu metafor kimi zaman bir gezegen, bir uydu kimi zaman geometrik şekiller, renkler ve dokular kimi zaman insanoğlunun habitatı olarak çalışmalara yansımıştır.

İçinde bulunduğumuz evren bize çoğu konuda ilham kaynağı olabilecek büyüklüğe ve sistematiğe sahiptir. Yazılım alanındaki bu görselleştirme çalışmaları örneklerinde de evrenin bu sistematiği ve hiyerarşisi ilham kaynağı olmuştur. Örneğin, kalıtım için gereken hiyerarşi güneş sistemindeki gezegenlerde bulunmuş ve kalıtım ağacının başındaki sınıflar gezegenlere, bu sınıflardan türetilen alt sınıflar ise uydulara benzetilerek modellenmiştir. Böylece yazılım tasarımı sırasında insanoğlunun yüzyıllardır alışageldiği bir sistematik kullanılarak problemlere yaklaşım kolaylaştırılmış ve yazılımı uygulanabilirlik açısından da güvence altına alınmıştır.

Aynı şekilde içerisinde yaşadığımız evler, mahalleler, şehirler de yazılım modellenmesinde birer metafor olarak gösterilerek aslında yazılımın insan yaşamı ile nasıl içe içe olduğunun anlaşılabilmesine de katkı sağlanmıştır.

Geometrik şekiller, renkler ve dokular hangi konu olursa olsun görselleştirme denilince akla gelen ilk materyaller olarak karşımıza çıkabilmektedir. Yazılım tasarımı sırasında ise UML sınıflarını görselleştirmede dörtgenler kullanılmış ve çalışmada karmaşıklığın önüne geçilmeye çalışılmıştır. Yine renk tonları arasındaki farklılıklar kullanılarak sınıfa bağımlılık oranlarının çok daha kolay anlaşılabilmesi amaçlanmıştır.

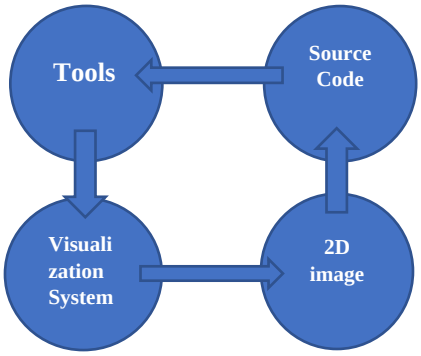
Yapılan bu görselleştirme çalışmaların her birinin kendi içerisinde sınırlamaları bulunmaktadır. Örneğin 3 boyutlu kutular kullanılarak yapılan modelde sınıf sayısı arttıkça görselleştirmenin yapılması ve bu üç boyutlu görsellik içerisinde gezinilmesi zorlaşacaktır. Yazılım tasarımı sırasında kullanılacak yazılımın gereklilikleri dikkate alınarak görselleştirme yapılmalı, her sorun kendine özgü görselleştirme metodunu da beraberinde getirebilmektedir. Yapılan çalışmalardan ve çıkarımlardan yola çıkarak yazılımcının kavrama yetisinin artması ve bilgilere dayalı karar verebilmesi gibi soyut görevlerde görselleştirmenin büyük etkisi vardır fakat literatürdeki bu çalışmaların büyük ölçekli yazılımları desteklemede ve görselliğe hızlı bir şekilde ulaşmada eksikliklerinin olduğu görülmektedir. Özellikle yapılan araştırmalar bütün kaynakların %50 ile %75'lik kısmının yazılımın bakımı için ayrıldığını [1,2] ayrılan bu kaynağın büyük bir kısmının ise yazılımın kavranabilmesi için harcandığını göstermiştir. Tersine mühendislik sayesinde yazılımlardan elde edilen yazılım metrikleri yazılımların anlaşılabilmesi, kontrol edilmesi ve yazılım kalitesinin iyileştirilmesine olanak sunmaktadır [4,5]. Ancak çok büyük ölçekli yazılımlar için elde edilen yazılım metriklerinden yazılım kalitesi ile ilgili çıkarımlar yapmak zaman alacaktır. Ayrıca birden fazla metriği aynı anda değerlendirmek gerektiğinde tablolar halinde sunulan metriklerden gerekli bilgileri elde etmek de kolay olmayacaktır [6].

Tüm bunlar göze alındığında her yazılıma kolayca entegre edilebilecek ve her yazılım için en uygun görselleştirme tekniğini sunabilecek bir sistemin önemi vurgulanabilmektedir.

3.Geliştirilen Yöntem

Literatürdeki çalışmalarda seçilen metaforların yazılım terimleri ve yazılım için gerekli hiyerarşi ile örtüşmesi sorunsuz olarak sağlanmış fakat büyük ölçekli yazılımları desteklemede eksikliği olduğu görülmüştür. Örneğin şehir metaforu kullanılarak oluşturulan şehir modeli çalışmasında; her bir sınıf kahverengi bir çubukla temsil edilmişti fakat kullanılan sınıf sayısı arttıkça görselleştirme işlemi zorlaşacak, çok büyük ölçekli yazılımlarda ise imkansız hale gelecektir. Yine benzer bir sorun 3 boyutlu kutuların kullanıldığı yazılım görselleştirme modelinde de yaşanmış ve yazılımın ölçeği büyüdükçe görselleştirme sırasında problemler yaşanabileceği öngörülmüştür. Geliştirilecek olan yöntemin yazılımın büyüklüğü ne olursa olsun etkili bir şekilde sonuç vermesi ve uygun görselleştirme tekniğini seçerek uygulayabilmesi beklenmektedir. Bu amaçla bir görselleştirme sistemi geliştirilmesi düşünülebilir. Bu sistem temel olarak 4 basamaktan oluşacak ve

barındırdığı karar-destek sistemi sayesinde hangi yazılıma hangi görselleştirme metodu uygulanabilirin kararını verebilecektir.



Literatürdeki en büyük sorun olan büyük ölçekli yazılımlara uygulanabilirlik, bu karar-destek sistemi sayesinde yazılım boyutuna en uygun görselleştirme modeli ile eşleşebilecektir. Bu çalışma yeni bir görselleştirme sistemi oluşturmak değil literatürde daha önce bulunan bir görselleştirme sistemine bir karar destek mekanizması ile geliştirmek hedeflenmiştir.

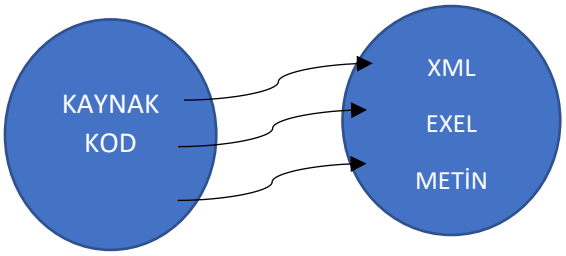
3.1 Source Code(Kaynak Kod)

Bu sistemin girdisi nesneye dayalı programlama dilleri olacaktır. Java, C++, C# gibi. Yazılımcının oluşturmuş olduğu kod alınıp bu sistemin birinci basamağına sokulduğu zaman yazılım kaynak kodlarından elde edilen metrikler veri olarak kabul edilecek ve birinci ünite bu basit metni XML ve Exel formatındaki metrikleri destekleyecek şekilde biçimlendirecektir. Elde edilen metrikler bir sonraki basamağa aktarılacaktır.

Bu ünite içerisindeki her bir sınıf için

- Paket
- Sınıf
- Metrik Tanımı
- Metrik Değeri

Elemanlarından oluşan bir küme oluşturur ve bir sonraki basamağa bu kümeyi iletir.



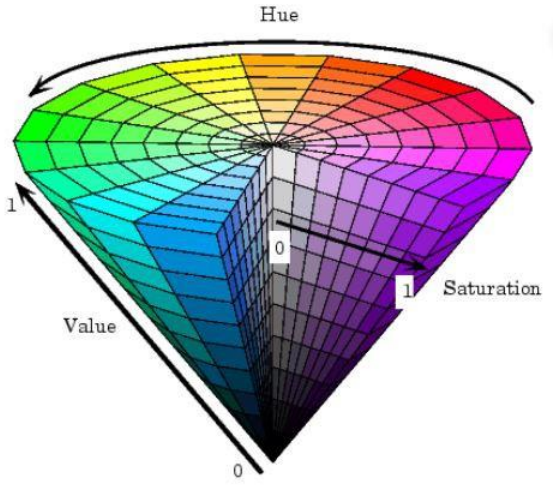
Kaynak Koddan Üretilen Küme

3.2 Tools(Araçlar)

Kaynak kod basamağından elde edilen metrikler bu aşamada çeşitli araçlar kullanılarak görselleştirme işlemine tabii tutulacak ve böylece her kaynak koda uygun görsel elde edilebilmesi hedeflenecektir. Bu aşamada renkleri kullanan toollar seçilmiştir.

Renk, ışığın değişik dalga boylarının gözün retinasına ulaşması ile ortaya çıkan bir algılamadır. Bu algılama, ışığın maddeler üzerine çarpması ve kısmen soğurulup kısmen yansıması nedeniyle çeşitlilik gösterir ki bunlar renk tonu veya renk olarak adlandırılır.[19]

Literatürde daha önce yapılmış olan bu çalışmada görselleştirme için kullanılan temel prensip metrik değerlerini farklı ışık frekansları olarak değerlendirip, birleşimlerine karşı düşen renkler toplamını bir görüntüde sunmaktır. Renkleri elde edebilmek ve tanımlanabilecek değişik renk uzayı modelleri oluşturulmuştur. RGB uzayı bütün renkleri Kırmızı, Yeşil ve Mavi renklerinin birleşimi olarak tanımlar.



Şekil 6

Bu aşamada bu modellerden RGB ve HSV renk uzaylarını kullanmıştır. Metrik aktarım ünitesinden gelen metrikler R,G,B ya da H,S,V renk bileşenlerine atanmıştır.

3.2.1 Normalizasyon

Dikkat edilmesi gereken önemli nokta ise metriklerin renk bileşenlerine atanması esnasında gerçekleşmesi gereken normalizasyondur.

Normalizasyon, veritabanlarında çok fazla sütun ve satırdan oluşan bir tabloyu tekrarlardan arındırmak için daha az satır ve sütun için alt kümelerine ayrıştırma işlemidir[20].

Amaç:

•Veri bütünlüğünün sağlanması yani gereksiz veri tekrarlarını önleyerek verilerdeki bozulmaları önlemek,

• Uygulamadan bağımsızlık, uygulama değişse bile veritabanının tutarlı olarak çalışması

•Performansı arttırmak, veri tekrarı en aza iner ve arama hızlı olur.

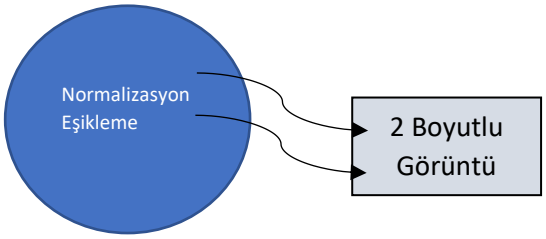
Metriklerin aldığı değerler ile renk bileşenlerinin alabileceği değerler uyumlu olmayabilir. Bazı metrikler [0,∞), bazıları sonlu aralıkta gerçel ya da tamsayı üretirken bazıları sınırlı bir aralıkta gerçek sayılar üretebilirler [21]. Diğer yandan RGB modelinde her bir bileşenin alabileceği değerler 0-255 arasında değişirken, HSV modelinde H, S, V bileşenlerinin alabileceği değerler sırasıyla [0,360], [0,100], [0,100] aralığında olabilmektedir. Görselleştirme araçları doğrusal normalizasyon metodu ile metrik değerlerini renk bileşenlerinin alabileceği değerlere uyumlu hale getirir. Kullanıcı farklı eşik değerleri belirleyerek bu değer aralıklarını kısıtlayabilir.

3.2.2 Eşikleme

Eşik: Yalnızca görüntü işleme için geçerli olan bir terim değildir. Herhangi bir alanda da eşik aynı anlama sahip olabilir. Genel olarak, herhangi bir fonksiyonun bir eşiği olabilir. Fonksiyon, eşik değerinin altında başka bir ifadeye sahipken, değerin üstünde bir başka ifadeye sahip olabilir.

Eşikleme: Verilen kaynak görüntüyü ikili(binary) görüntüye çevirmek isterseniz ve görüntü üzerindeki gürültüleri azaltmak veya üzerinde nesne belirlemek gibi (örneğin fotoğraf/ video üzerinde bir dergiyi ekranda özel olarak belirleme) farklı amaçlarla kullanabilirsiniz. Giriş olarak verilen görüntü üzerinde uygulanan eşikleme tipine bağlı olarak, pikselleri verilen eşik değerine göre siyah ya da beyaz olarak günceller.

Bu aşamada normalizsayon kadar eşikleme de oldukça önemlidir.



Bu aşamadan sonra oluşan görseller bir sonraki aşamaya aktarılır.

3.3 Visualization System(Görselleştirme Sistemi)

Bu aşamaya önceki basamaklardan elde edilen görseller, gerekli araçların kullanılıp işlenmesinden sonra iletilirler. Bu kısım için literatürdeki çalışmadan farklı olarak bir karar-destek sistemi kullanılmıştır.

Yapılan kaynak taramaları sonucu en büyük eksikliğin büyük ölçekli yazılımlarda görselleştirme sisteminin uygulanırken sıkıntı çıkarması olduğu görülmüş ve bu duruma bir çözüm olarak bu basamakta bir karar destek sistemi kullanılarak hangi yazılım için hangi görselleştirme modelinin kullanılabileceğine en doğru ve en hızlı biçimde ulaşılmak istenmiştir.

3.3.1 Karar-Destek Sistemi

Bir karar destek (KDS, karar verme yazılımı olarak da bilinir- DMS), iş veya kurumsal karar verme faaliyetlerini destekleyen, genellikle sıralama, sıralama veya alternatifler arasından seçim yapmaya dayanan bir bilgisayara dayalı bilgi sistemidir. Karar destek sistemleri bir organizasyonun yönetim, operasyon ve planlama seviyelerine hizmet eder ve insanlara hızla değişen ve önceden kolayca belirlenemeyen sorunlar hakkında kararlar vermelerine yardımcı olur. Bunlar yapılandırılmamış ve yarı yapısal karar problemleri olabilir. Karar destek sistemleri tamamen bilgisayarlı, insan gücüyle veya her ikisinin birleşimi olabilir. Karar destek sistemleri bilgiye dayalı sistemleri içerir. Düzgün tasarlanmış bir karar destek sistemi, kararı belirleyip çözmek ve karar vermek için karar vericilere ham veri, belge ve kişisel bilgi birikiminden veya kişisel bilgi birikiminden yararlı bilgiler derleyebilmesine yardımcı olmak için tasarlanmış interaktif bir yazılım tabanlı sistemdir[22].

Bu çalışmada ise yazılım tasarımı görselleştirmede de bu sistemlerden faydalanılarak ilgili yazılıma hangi görselleştirme modelinin daha uygun olacağına karar verilmede kullanılabileceği düşünülmüştür. Görselleştirmek istenilen yazılımda kullanılacak sınıflar, metotlar, değişkenler, yapılar, nesneler için uygun görselleştirme modeli yazılımın büyüklüğüne bağlı olarak karar destek sistemi tarafından seçilecektir.

3.4 İki Boyutlu Görüntü

Görselleştirme sisteminin çıktısı olan iki boyutlu görüntü, aşağıdaki özellikleri yardımıyla analiz edilir.

- ***Odak****:* Bu özellik ortaya çıkan görsellik içerisinde yazılımdaki tüm sınıflarını birden görme ile tek bir sınıfı görme arasındaki tüm düzeylere odaklanabilme imkânı sunar.
- ***Bilgi:*** Sistem görsellik üzerinde dolaşırken üzerinde bulunan her bir beneğin karşı düştüğü sınıfı, sınıfın içinde bulunduğu paketi, sınıfın yer aldığı kaynak kodunu ve bu piksel değerini oluşturan renk bileşenlerinin değerleri ile metrik değerlerini gösterir.
- ***Sıralama:*** Analiz esnasında görüntü istenilen metrik değerine göre sıralanabilir. Örneğin DIT ve NOM metriklerinin oluşturduğu bir görsellikte DIT metriğine göre yapılan sıralama sınıfları kalıtım seviyesine göre dizerek sınıfların sahip oldukları metot sayılarının kalıtım seviyelerine göre dağılımını görebiliriz.
- ***Filtreleme:*** Birden fazla metriğin renk bileşenlerine atanmasıyla oluşan görsellikte herhangi bir bileşenin etkisi Açılıp kapatılabilir. Böylelikle bileşenin tek başına ya da farklı ikili kombinasyonlarla analiz edilebilir.
- ***Görüntü saklama ve karşılaştırma:*** Elde edilen sonuçlar son aşamada saklanabilir ve yazılımın farklı sürümlerindeki sonuçlarla ya da farklı yazılımların görsellikleriyle karşılaştırılabilir.

4.Yapılan İncelemeler Sonucu Yazılım Tasarımı için Tasarlanan Model

Bu kısımda ilk olarak görselleştirme ölçütleri anlatılmış ve sonrasında Java dili için bir apartman dairesi modeli geliştirilmiştir.

Metafor; bir durumu, sorunu, vakayı başka bir şekilde ifade etmek için kullanılır. Bir şeyi başka şeyle anlatmaya, benzetmeye yarayan mecazlardır. Mecaz sanatı sayesinde mesaja canlılık, derinlik ve güçlü bir anlam kazandırılır[23].

Bilimsel ya da teknik konuları anlamakta zorlanmamız, konsantrasyonumuzu çabuk kaybetmemiz, konuların zorluğundan çok anlatım dilinin soyut olmasındandır. Dil ne kadar soyut olursa anlaşılması o kadar zorlaşır. Bu noktada yaptığımız görselleştirme çalışmalarının konuları anlama ve kavrama yeteneklerimizi kolaylaştırması amaçlanmaktadır. Yapılan çalışmalar da gösteriyor ki beyne gönderilen bilgilerin %90'ı görseldir ve beyin görselleri metne kıyasla 60.000 kat daha hızlı işler.[3]

Yazılım konusunda da görsellik, modelleme, metaforlar tasarım aşamasında yazılımın kavranabilmesi, uygulanabilmesi açısından kolaylık sağlayabilmektedir. Yapılan çalışmalar gösteriyor ki bütün kaynakların %50 ile %75'lik kısmının yazılımın bakımı için ayrıldığını [1,2] ayrılan bu kaynağın büyük bir kısmının ise yazılımın kavranabilmesi için harcanıyor. Tersine mühendislik sayesinde yazılımlardan elde edilen yazılım metrikleri yazılımların anlaşılabilmesi, kontrol edilmesi ve yazılım kalitesinin iyileştirilmesine olanak sunmaktadır [4,5]. Fakat çok büyük ölçekli yazılımlar için elde edilen yazılım metriklerinden yazılım kalitesi ile ilgili çıkarımlar yapmak zaman alacaktır. Bu gibi zaman alacak durumlarla karşı karşıya kalmamak adına yazılım tasarımında görselleştirme yoluna gidilmiş ve yazılımlar için uygun metaforlar kullanılarak yazılımlar modellenmiştir.

Modellemeni faydaları:

- Ele alınan yazılım problemin çözümünün bilimsel temellere oturmasının sağlanması,
- Konuların ve araştırılan bulguların daha basit hale indirgenmesiyle kavranabilirliğinin arttırılması,

- Hızlı veri aktarımlarının ve hızlı sonuçların alınarak zamandan tasarruf sağlanması

şeklinde sıralanabilir.

4.1Görselleştirme Ölçütleri

***Sınırlar [31]*:** Görsel modelin geometrik ve görsel sınırları açıkça anlaşılabilmelidir. Görsel modelin ve veri elemanlarının kapladığı alan belli olmalıdır.

Veri Yoğunluğu [31, 18, 33, 34]: Belli bir alandaki veri elemanı sayısına dikkat edilmelidir. Veri elemanlarının çok sıkışık veya çok dağınık olması durumunda okunabilirlik azalmaktadır. Bu ölçüt bazı kaynaklarda, "Bilgi yoğunluğu" [32] veya "Çoğu veri en küçük alana konulmalıdır" [33, 34] olarak tanımlanmaktadır.

Veri Boyutu [31]: Görsel model üzerinde eş zamanlı olarak gösterilen veri boyutu sayısına da dikkat edilmelidir.

İlişkili Bilgilerin Gösterimi [31]: Görsel modelde veri nesnesiyle ilişkili farklı bilgilerin anlamlı ve uygun olmasına dikkat edilmelidir. Çok fazla ya da gereksiz ilişkili bilgi verilmesi karmaşıklığı arttırmaktadır.

Mantıksal Sıra [31]: Veri elemanları, karakteristik özelliklerine göre mantıksal bir sırada bulunmalıdır. Bu, veri elemanlarına kolay ulaşılmasını sağlamaktadır. Renklerin mantıksal bir sırası bulunmaz. Her rengin önceliği kişiye göre değişebilmektedir. Bu yüzden veri elemanları renklere göre sıralanmamalıdır. Bu ölçüt bazı kaynaklardaki, "Renkten bir okuma sırası beklenmemelidir" [33, 34] ölçütünü kapsamaktadır.

Örtme [31]: Görsel sunumun belli bir görüntü alanı bulunduğundan veri elemanı sayısı arttığında birbirinin üzerine gelebilmektedir. Bu durumdan olabildiğince kaçınılmalıdır.

Detayların Gösterimi [31]: Detaylara sahip olan veri elemanları açıkça anlaşılabilmelidir.

Bağlam Referansı [31]: Herhangi bir veri elemanının bağlamla olan referansı devamlı belirgin olmalıdır.

Bilgi Adresleme [31, 33, 34: Veri elemanlarının doğru görsel elemanlara uygun şekilde dönüştürülmesi gerekmektedir. Dönüştürme sonunda verinin, karakteristiğini kaybetmemesi gerekmektedir. Bu ölçüt bazı kaynaklarda, "Metinler ilişkiliyse grafiklerle bütünleştirilmelidir" [33, 34] veya "Grafik boyutsallığında veri korunmalıdır" [33, 34] olarak tanımlanmıştır.

Gerçekçi Teknikler [13, 16]: Veri elemanlarının dönüşümünde gerçek dünyadaki kavramların kullanılması veri elemanlarının algılanmasına yardımcı olur. Bu ölçüt bazı kaynaklarda, "Sistem ile gerçek dünya arasındaki uyum" [16] olarak tanımlanmıştır.

Görselin Oluşturulma Zamanı [31]: Görsel sunumun bir kısmının veya tamamının oluşturulması için gereken süre olabildiğince azaltılmalıdır.

Görsel/Konumsal Uyum [31]: Görsel elemanların konumsal değişimi, görsel elemanın karakteristiğine uygun olmalıdır. Bu bilginin algılanmasının kolaylaştırır.

Görsel Değişkenin Büyüklüğü [23, 25]: Görsel sunumdaki her görsel eleman anlaşılabilir derecede büyük olmalıdır. Bu ölçüt bazı kaynaklarda, "Görsel değişkenin yeterli büyüklükte olduğundan emin olunmalıdır" [33, 34] olarak tanımlanmıştır.

Renk ve Büyüklük Algısı [33, 34]: Renkli nesnelerin algılanması, nesnelerin büyüklüğüne göre değişiklik gösterebilir. Örneğin, kırmızı renkteki bir görsel eleman, diğer benzer görsel elemanlardan daha küçük boyutta olursa, önemli bir veri elemanı olduğu algılanamayabilir. Bu ölçüt bazı kaynaklarda, "Renk algısı renkli nesnelerin büyüklüğüne göre değişir" [33, 34] olarak tanımlanmıştır.

Yerel Kontrast [23, 25]: Görsel sunumun görüntülendiği donanım, renk ve griliklerde değişkenlik yaratarak, farklı algılar oluşmasına sebep olabilir. Renklerin olabildiğince zıt seçilmesi bu oluşumu engelleyebilir. Bu ölçüt bazı kaynaklarda, "Yerel kontrast, renk ve grilik algısını etkiler" [33, 34] olarak tanımlanmıştır.

Renk Körlüğü [23, 25]: Evrensel kullanılabilirlik düşünüldüğünde, renklerin tek başına ayırt edici olarak kullanılmaması gerekmektedir. Bu ölçüt bazı kaynaklarda, "Renk körlüğü olan insanları düşün" [33, 34] olarak tanımlanmıştır.

***Ön Dikkat İşlemi [23, 25*]:** Ön dikkat işlemi, önemli bilgilere dikkat çekmek için farklı görsel değişkenlerin önceden kullanılmasıdır. Bu görsel değişkenlerin özenle seçilerek kullanılması gerekir. Bu ölçüt bazı kaynaklarda, "Ön dikkat işlemi, ön dikkat işlemindeki görsel değişkenlerin akıllıca görüntülenmesi veya kullanılmasıyla fayda sağlar" olarak tanımlanmıştır.

Niceliksel değerlendirme [33, 34]: Sayısal değerleri bulunan veri elemanlarının konumu veya büyüklükleri değerlerine göre değiştirilmelidir. Bu ölçüt bazı kaynaklarda, "Niceliksel değerlendirme konum veya büyüklükte çeşitlilik gerektirir" [33, 34] olarak tanımlanmıştır.

Gestalt ilkeleri [33, 34]: Gestalt ilkeleri insanın algılama boyutundaki ilke ve sistemlerini ortaya koymaktadır. Kullanıcıda doğru bir algı yaratabilmek için, görsel sunumda bu ilkeler göz önünde bulundurulmalıdır.

4.2 Görsel Model ve Etkileşim Ölçütleri

Yardım ve Belgelendirme [35]: Kullanıcı ihtiyaç duymasa bile, görsel sunumdaki nesneler hakkında bilgilere istediğinde ulaşılabilmelidir. Görsel sunum ile kurulabilecek etkileşimler adım adım açıklanmalıdır. Açıklamaların çok uzun olmamasına dikkat edilmelidir.

Tutarlılık ve standartlar [35, 32]: Görsel sunum üzerindeki nesnelerin tutarlı olması gerekir. Nesnenin tanımı veya görsel öğesi her zaman aynı olmalıdır. Görsel öğe ile tanım birbirine uygun olmalıdır. Etkileşim mekanizmalarında kullanılan öğelerin de tutarlı olması gerekir. Tutarlılığı sağlamak için standartlar göz önünde bulundurulmalıdır.

Memnuniyet [39, 32, 36]: Kullanıcının görselleştirme ile ilgili düşüncelerini belirtir. Kullanıcının görselleştirme üzerindeki hâkimiyeti memnuniyeti arttıran bir faktördür. Bu ölçüt bazı kaynaklarda "Kullanıcı tecrübesi" [18] olarak tanımlanmıştır.

Açıklık ve anlaşılabilirlik [37, 38, 35]: Kullanıcı görselleştirme aracılığı ile aktarılmak istenen bilgiye, geçmiş bilgilerine ihtiyaç duymadan ulaşabilmelidir. Bu ölçüt bazı kaynaklarda, "Okunabilirlik" [9] veya "Hatırlama yerine tanıma" [16] olarak tanımlanmıştır.

Özlülük [38]: Görselleştirmede asıl aktarılmak istenen bilgiler en öz halleriyle verilmelidir. Detaylar, kullanıcı isteği ile görüntülenmelidir. 237

Öğrenme kolaylığı [37, 39, 36]: Kullanıcının bilgi görselleştirmeyi ilk karşılaşmasında kolayca etkin bir şekilde kullanabilmesidir. Bu ölçüt bazı kaynaklarda, "Öğrenilebilirlik" [15, 20] olarak tanımlanmıştır.

Ustalaşma kolaylığı [37]: Bilgi görselleştirmenin hızlıca tüm özellikleriyle etkin bir şekilde kullanılabilmelidir.

Hatırlanabilirlik [39, 36]: Anlık durum hakkında yorum yapabilmek için daha önce yapılan sorgular hatırlanabilmelidir.

Kullanma/anlama kolaylığı [37]: Kullanıcı çok az bir zihinsel çaba ile bilgi görselleştirmeyi kullanabilmelidir.

Etkinlik [37, 39, 35, 36]: Öğrenme aşamasından sonra kullanıcı hızlı bir şekilde görevleri başarabilmelidir.

Estetik [37, 40, 41, 16]: Görselleştirme eğlenceli olmalı ve göze hoş görünmelidir. Çok fazla görsel stil kullanmaktan kaçınılmalıdır [10]. Etkileşim anında sert durum geçişlerinden kaçınılmalıdır. Bu ölçüt bazı kaynaklarda, "Estetik ve sade tasarım" [16] olarak tanımlanmıştır.

Tüm bunlardan yola çıkarak Java dili kullanılan bir yazılım için görselleştirme çalışması yapmak istiyorum ve metafor olarak bir apartman dairesini kullanarak bu görselleştirme işlemini gerçekleştiriyorum.

Yazılımın tasarımına geçmeden önce Java programlama dilinin özellikleri şöyle sıralanabilir[24]:

- Nesne yönelimli bir dildir. Bir java programında olabilecek her şey ya nesnedir ya da nesnenin bir parçasıdır.
- Namespace kavramı paketlerle sağlanır.
- Güvenli bir dildir. Hiçbir virüs bir Java programına bulaşamaz. Kötü niyetli bir program, bir işlemi eğer izin verilmediyse yapamaz.

- Dinamiktir. Java'da bir programla kullandığı birimlerin (kütüphaneler, modüller veya sınıfların) birbirine bağlanması çalıştırma anında yapılır.
- Taşınabilirdir. Java programları her ortamda aynı veya benzer bir şekilde çalışır.
- Basit bir dildir. Kendisine yakın güçteki dillerin en basitidir.
- Sınıflardan oluşur.
- Yorumlanabilir bir dildir. Yani bir Java programının komutları, çalışırken Java Virtual Machine (JVM) tarafından makinenin anlayacağı formata çevrilir. Bunun avantajı bir programın kullanıldığı standart kütüphanelerin programla birlikte taşınması zorunluluğunu ortadan kaldırmasındır.
- Dağıtıktır yani birden fazla bilgisayarda çalışan programların birbiriyle uyumlu çalışabilmesidir.
- Sağlamdır. Başka dillerin aksine çalışma esnasında bir Java programı "Bir hata oldu!" deyip çökmez.
- Mimarilere yansızdır. Java'da yazılan bir program hemen hemen bütün işletim sistemlerinde hiç değiştirmeye gerek duymaksızın çalışır.
- Yüksek başarımlı bir dildir. Diğer dillerde olmayan birçok özelliğe sahip olmasına rağmen, Java'da bunun için fazla bir performans kaybı yoktur.
- Java dili çok kanallıdır. Çok kanallılık (multi threading) bir programın aynı anda birden fazla işlemi yürütebilmesi demektir.

➢ *Apartman Dairesi Modeli*

Bu çalışmada Java yazılım tasarımı için şekil 7'deki apartman dairesi metaforu kullanılmış ve bir daire modeli oluşturulmuştur.

Şekil 7

Dairenin her bir odası Java'daki sınıfları temsil etmektedir. Her odada yaşayan bireyler bu sınıflara ait nesneleri, bireylerin yaşamsal faaliyetleri metotları, dairenin tamamı ise Java paketlerini temsil etmektedir. Odalarda bulunan bireyler aynı çatı altında oldukları için birbirleri ile iletişim kurabilmektedir. Bu durum Java'da aynı paket altında bulunan sınıf metotlarının birleri ile etkileşim halinde olabileceğine benzetilmiştir. Java dili taşınabilir bir dildir bu özellik ise bu apartman dairesinde yaşayan bir ailenin istedikleri zaman başka bir şehirde başka bir sitede başka bir dairede yaşayabilecekleri durumuna benzetilmiştir. Java zararlı yazılımlara karşı güvenli bir dildir. Dairede yaşayan insanlar sitede bulunan güvenlik, alarmlar ve güvenlik kameraları gibi sistemler sayesinde kendilerini hırsız vb kötü niyetli insanlardan koruma altına alırlar. Java mimarilere yansız bir dildir yani java dilinde yazılmış bir program hemen hemen her işletim sisteminde değiştirilmeye gerek duyulmadan çalışabilir, bu durum ise apartman dairesinde yaşayan bir aile başka bir daireye geçse de hayatlarını sürdürebilirler.

4.1 Deney ve Gözlem

Java dili kullanılarak oluşturulacak görselleştirmenin anlaşılabilirliği ve fayda sağlayabilirliği açısından deney ve gözlemin yapıldığı kısımdır.

Örnek olarak Java dilini öğrenmek isteyen ve bu dil ile bir yazılım tasarımı yapmak isteyen birisi için bu görselleştirme yazılımın kavranabilmesi ve uygulanabilmesi açısından faydalı olacaktır. Bunu gözlemleyebilmek için 16 yaşındaki lise öğrencisi kardeşim üzerinde bir deney yaptım. İlk olarak Java programlama dilini yeni yeni öğrenmeye başlayan 16 yaşındaki kardeşime, herhangi bir görsel tasarım olmadan Java dilini ve özelliklerini anlattım daha sonra ise oluşturduğum bu görsel tasarımı göstererek ve yaptığım benzetmeler ile tekrar anlattım. Her iki aşamadan sonra anlaşılabilirliği test etmek için sorular yönelttim ve görselleştirme yaparak anlattığım kısımda daha doğru dönütler aldım.

Bu beklenen bir sonuç olarak yansıdı aslında çünkü yapılan çalışmalar da gösteriyor ki beyne gönderilen bilgilerin %90'ı görseldir ve beyin görselleri metne kıyasla 60.000 kat daha hızlı işler.[3] Bu noktalar, bilgiyi analiz etmek ve iletmek için veri görselleştirmesi kullanmanın önemini güçlü bir biçimde ortaya koymaktadır.

4.2 Neden Apartman Dairesi Metaforu

Evlerimiz hepimizin konfor alanlarıdır. Kendimizi en rahat hissedebileceğimiz, doğduğumuz andan beri içerisinde bulunduğumuz, barınma gibi temel ihtiyacımızı karşılayabilecek, aile gibi soyut ve toplumun en temel kavramı olan birimin derinden hissedildiği çatıdır. Tüm bunlar düşünüldüğünde neden ev metaforu kullanarak bir yazılım modeli oluşturulabilirin cevabı ortaya çıkıyor.

Ev modeli insanların doğumundan süregelerek alışık olduğu içerisinde hayatlarını sürdürdüğü, konforuna ve sistematiğine hakim olduğu bir metafor olduğu için bu model kullanılarak konunun anlaşılmasının daha kolay olabileceği ve hiç yazılım bilmeyen birinin dahi bu görselleştirme ile bahsedilen Java dili hakkında rahatça bilgi sahibi olabileceği düşünülmüştür. Bu metafor seçilirken herkesin hakkında bilgi sahibi olması ve herkesin yaşadığı ev içerisinde işleyişe hakim olması göz önünde bulundurulmuş, yapılan seçime etken olmuştur.

5.Yapılan Çalışmanın Karşılaştırılması

Literatürde bulunan görselleştirmenin yazılım tasarımındaki etkisi konuları ve bu konu ile ilişkili konular, ilgili materyalleriyle taranmış ve yapılan çalışma ile ilgili karşılaştırılıp benzerlikleri, farklılıkları ve paralellikleri hakkında tartışıldığı kısımdır.

5.1 Benzerlikleri

Yapılan bu çalışmada Java yazılımı için görselleştirme modeli olarak bir apartman dairesi kullanılmış ve bu programlama dili ile daire, daire hayatı arasında benzetimler yapılarak anlaşılabilirliğin artması amaçlanmıştır. İnsanlık için alışılagelmiş ve herkes tarafından bilinen bir metafor seçimi yapılmasına dikkat edilmiştir.

Literatürdeki diğer yazılım modellemelerine bakıldığında

- Şehir Modeli
- Evren Modeli

şeklinde yapılan çalışmalarda seçilen metaforlar şehir, araçlar, gezegenler, yıldızlar, ay şeklinde insanlığın yüzyıllardır alışageldiği ve anlaşılması kolay olabilecek modelleme metaforları olarak seçilmiştir. Yapılmış olan bu çalışmadaki modelleme metaforunun seçilmesindeki en önemli etken ise insanların alışageldiği ve anlaşılabilmesi kolay olabilecek bir metafor seçebilmekti. Literatürdeki şehir modeli ve evren modeli çalışmalarında da anlaşılabilirlik ve alışılagelmişlik benimsenmiştir. Bu bakımdan yapmış olduğum çalışmadaki bu husus literatürdeki çalışmalarla benzerlik göstermektedir.

Faktörler	Şehir Modeli ve Evren Modeli	Apartman dairesi modeli
Renk	✓	✓
Anlaşılabilirlik	✓	✓
Metafor kullanımı	✓	✓
Alışılmışlık	✓	✓
Geometrik Şekiller	✓	✓
Doku		

Apartman dairesi modeli ve literatürdeki diğer çalışmaların ortak özellikleri gösteren Tablo 1

Geliştirilen görselleştirme sisteminde ise daha önce var olan bir çalışmaya karar destek sistemi eklenmiş, diğer safhalar olan analiz ünitesi, görselleştirme ünitesi ve kaynak koddan metrik aktarım ünitesi daha önce yapılan çalışmadaki gibi kullanılmıştır.

Sonuç olarak; daha önce yapılan çalışmalardan çok uzaklaşılmamış, incelenerek eksik görülen kısımlar tamamlanmaya çalışılmıştır.

5.2 Farklılıkları

Literatür taraması yapıldığında karşılaşılan modelleme çalışmalarından

- 3 Boyutlu kutuların modellenmesi
- Renk ve dokunun birlikte kullanılarak modellenmesi

çalışmalarında, iki boyutlu bir görsellik sunulurken, insanın görsel algılama yetisinin karmaşık görsellikleri de rahatça kavrayabileceği varsayılarak doku ve renk de birlikte kullanmıştır.

Şehir modeli ve evren modelinden farklı olarak; insanların görsel algılama yetisinin sadece kolay anlaşılabilir basit metaforlarla sınırlı olmadığını ve bu yüzden de karmaşık görseller de kullanılabilerek modelleme yapılabileceği düşünülmüştür. Bu bağlamda 3 boyutlu kutularla yapılan modellemede kutuların en, boy ve yükseklikleriyle birlikte dizilişlerinde

yaptıkları açı ve renkleri de ayrı birer metriği temsil etmek üzere karmaşık bir şekilde kullanılmıştır.

Bir diğer örnek model olan Yazılım Evreni modelinde de yazılımın farklı sürümlerinden elde ettiği 20 farklı metriği yazılımın 7 farklı sürümü için kiviat diyagramı kullanılarak bir arada verilmiştir.

Yapılan her iki modelleme çalışmasında da basitlik ve kolay anlaşılırlık yerine kompleks yapılar kullanılarak modellemeye gidilmiştir. Benzerlikleri başlığı altında da bahsedildiği gibi bu çalışmada yapılan apartman dairesi modellemesinde ise temel alınan hususlar kolay anlaşılabilirlik ve alışılagelmişlik olmuştur. Bu bağlamda bu çalışma ile literatürdeki yazılım evreni modeli ve 3 boyutlu kutular kullanılarak yapılan modelleme arasındaki farklılıklar olarak karşımıza çıkmaktadır.

Faktörler	Yazılım Evreni Modeli ve 3 boyutlu kutu modeli	Apartman Dairesi Modeli
Koplekslik	✓	
Metafor kullanımı	✓	✓
Alışılmışlık		✓
Renk ve doku	✓	
3 boyut	✓	

Apartman Dairesi Modeli ve diğer iki model karşılaştırılmıştır Tablo 2

Yapılmış olan görselleştirme sisteminde ise literatürdekinden farklı olarak bir karar-destek sistemi eklenmiştir çünkü yapılan çalışmalar incelendiğinde en büyük sorunun yapılan görselleştirme işleminin büyük ölçekli yazılımlara uygulanamaması olduğu görülmüş ve buna bir çözüm olarak karar-destek sistemi ile hangi yazılıma hangi görselleştirme sisteminin uygun olacağına karar vermek için bir karar destek programı kullanılması gerektiği düşünülmüştür. Bu bağlamda bakıldığında yapılan çalışma ile literatürdeki çalışmalar arasındaki farklıları gözlemleyerek neden bu değişikliklere ya da eklenemelere gidildiğini görmekteyiz.

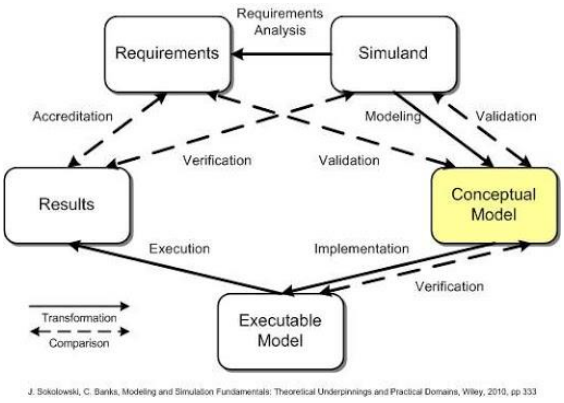
Faktörler	Literatürdeki Görselleştirme Sistemi	Bu Çalışmadaki Görselleştirme Sistemi
2D Görüntü	✓	✓
Analiz	✓	✓
Aktarım	✓	✓
Karar-destek Sistemi		✓
Normalizasyon	✓	✓
Eşikleme		✓

Literatürdeki çalışmada bulunan görselleştirme sistemi ve bu çalışmada yapılan görselleştirme sistemi karşılaştırılmıştır Tablo3

5.3 Paralellikleri

Literatürdeki yapılmış yazılım modelleme çalışmalarında ve bu çalışmada asıl amaç yazılımcının yazılımı görsellik sayesinde daha kolay anlaması ve bundan sebep zaman kayıplarının en aza indirilmesi olmuştur. Yapılan araştırmalar bütün kaynakların %50 ile %75'lik kısmının yazılımın bakımı için ayrıldığını [1,2] ayrılan bu kaynağın büyük bir kısmının ise yazılımın kavranabilmesi için harcadığını göstermiştir. Tersine mühendislik sayesinde yazılımlardan elde edilen yazılım metrikleri yazılımların anlaşılabilmesi, kontrol edilmesi ve yazılım kalitesinin iyileştirilmesine olanak sunmaktadır [4,5]. Ancak çok büyük ölçekli yazılımlar için elde edilen yazılım metriklerinden yazılım kalitesi ile ilgili çıkarımlar yapmak zaman alacaktır. Ayrıca birden fazla metriği aynı anda değerlendirmek gerektiğinde tablolar halinde sunulan metriklerden gerekli bilgileri elde etmek kolay olmayacaktır[6]. Bu sorunun etrafında çözüm yoluna gidilmeye çalışılmıştır.

Metafor kullanmanın görselleştirmedeki önemi de göz önüne alınmış ve modelleme sırasında

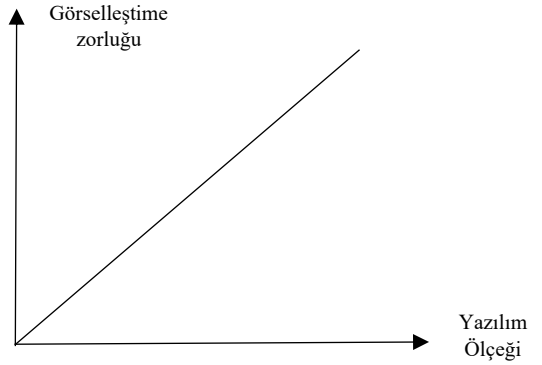


metafor kullanılarak anlaşılabilirliğin en yüksek seviyede olmasına gayret edilmiştir. Bu konuda literatürle paralellik sağlanmıştır.

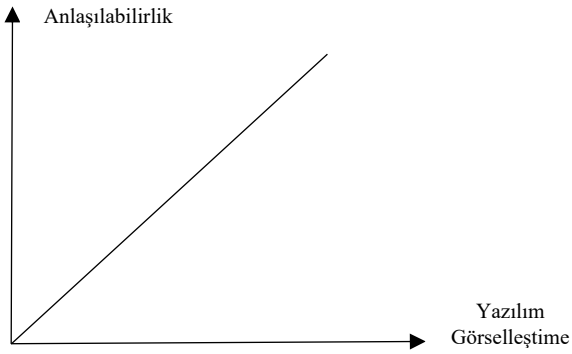
6.SONUÇ

6.1 Teorik Olarak

Bir bilgi görselleştirme yöntemi ya da tekniği; özenle hazırlanmış bir kavrayışı, iç görüler elde edilmesini ve deneyimlerle iletişime geçilmesini sağlayan bilgiyi tasvir eden sistematik, kural tabanlı, dışsal, kalıcı ve grafiksel bir sunumdur [27]. Görselleştirme sürecinde mutlaka bir görselleştirme tekniği kullanılma zorunluluğu yoktur. Fakat veri kümesi karmaşıklaştıkça bir tekniğin kullanılması ihtiyacı da artmaktadır. Literatürde tanımlanmış her bir görselleştirme tekniğinin uygun olduğu en az bir veri kümesi bulunmaktadır. Literatürde bugüne kadar tanımlanmış 100'den fazla görselleştirme tekniği bulunmaktadır [27]. Bu tekniklerden de faydalanarak yazılım modellemeye gidilmiş ve yazılım projeleri sırasında yazılımın anlaşılması konusunda karşılaşılabilecek sorunların önüne geçebilmek amaçlanmıştır. Varılan sonuç ile görselleştirmenin yazılım üzerindeki etkisinin olumlu yönde olduğu noktasında toplanmıştır. Fakat yazılımın ölçeği büyüdükçe modellemenin yapılmasının zorlaştığı kanısına da varılmıştır. Teorik olarak varılan sonuçları tablolara dökmek gerekirse:



Yazılım ne kadar büyük ölçekli olursa görselleştirtirme o kadar zorlaşıyor. Modelleme uzun uğraşlar zaman alabiliyor. Tabloda da görüldüğü üzere yazılım büyüklüğü ve görselleştirme zorluğu doğru orantılı ilerliyor.



Yapılan bu çalışma sonucunda yazılımda görselleştirme ile kolay anlaşılabilirlik doğru orantılı olsuğu kanısına varılmıştır.

6.2 Uygulama Olarak

Uygulama kısmı için bir yazılım görselleştirme modeli oluşturuldu. Java programlama dili kullanılarak yazılmış bir yazılım için apartman dairesi modeli tasarlandı. Java dilinin özellikleri, terminolojisi ve işleyişi göz önüne alınarak hazırlanan modelde, java ile yazılmış bir yazılımın daha iyi anlaşılabilmesi için modellemede anlaşılabilirlik ve basitlik göz önüne alındı. Seçilen metafor olan dairede ise herkes tarafından bilinen ve herkesin içerisinde yaladığı bir nesne seçildi ki hiç yazılım bilgisi olmayan birisi için bile anlaşılabilir olması hedeflendi.

Oluşturulan modelleme 16 yaşında erkek bir lise öğrencisi üzerinde test edildi ve olumlu dönütler alındı. Yapılan deneyin ilk aşamasında yazılım konusunda bilgi sahibi olmayan 16 yaşındaki lise öğrencisine, herhangi bir modelleme olmaksızın Java dili ve Java dili üzerinden yazılmış küçük bir yazılımın anlatımı yapıldı. İkinci aşamasında ise oluşturulmuş olan apartman dairesi modellemesi kullanılarak java dili ve bu dille yazılmış olan küçük bir yazılım gösterilerek tekrar anlatıldı. Her iki anlatımdan sonra alınan dönütlere göre yazılım için görselleştirme yapılan ikinci aşamadan sonra daha doğru ve net cevaplar alınmıştır. Bu uygulamadan sonra yapılan çalışmada görselleştirmenin yazılımın üzerindeki etkinin su götürmez bir gerçek olduğu görülmüştür.

6.3 Literatüre Kazandırılan

Konu hakkında literatür taraması yapıldığında görülmüş olan modellemeler:

- Şehir Modellemesi
- Evren Modellemesi

- Yazılım Evreni Modellemesi
- 3 Boyutlu Kutu Modellemesi

şeklinde sıralanmaktadır.

- ***Apartman Dairesi Modeli***

Tüm bu modellemelere bakıldığında yazılım terimlerinin, yazılım hiyerarşisinin, işlevlerinin daha net anlaşılabilmesi için kurgulanmış olan modeller olduğu görüldü. Şehir modellemesi hariç diğer tüm modeller uygulamaya geçirilmiş eksikleri ve gereklilikleri fark edilmiştir. En büyük sorunun büyük ölçekli yazılımlarda modelleme uygulanmasının zor olduğu kanısına varılmıştır.

Literatür taraması sonucu herhangi bir yazılım dili baz alınarak yapılmış bir modellemenin olmadığı görülmüş ve bundan hareketle Java yazılım dili seçilerek Java dili özellikleri de belirtilerek içerisinde yaşadığımız daireye benzetip modelleme yapılmıştır.

Bu çalışmada modelleme olan **"Apartman Dairesi Modeli"** bu konu için literatüre kazandırılmış bir yazılım modeli olmuştur.

- ***Görselleştirme Sistemi***

Görselleştirme işlemi için bir yazılım görselleştirme sistemi gerekliliği düşünülmüş ve yapılan literatür taraması sonucu daha önce yapılan bir yazılım görselleştirme sistemi incelenmiştir [25]. İncelenen bu sistemde geliştirmeler yapılarak sisteme bir karar-destek sistemi eklenmesi gerektiği düşünülmüştür. Geliştirilen bu karar-destek sistemi sayesinde hangi yazılım için hangi hangi modelleme tekniğinin daha uygun olabileceği konusunda daha istikrarlı kararlar verilebileceği ve modellemeyi uygulama kısmında geri dönüşü olmayan hatalar yapılmasının önüne geçilmek istenmiştir.

Bu çalışmada daha önce yapılmış olan "**Görselleştirme Sistemi**" geliştirilerek bir karar destek sistemi eklenmiş ve literatüre kazandırılmıştır.

6.4 Günlük Hayata Kazandırılan

Tüm bu yapılan çalışmalar sonucu görülmüştür ki görselleştirmenin yazılım tasarımı üzerindeki etkisi göz ardı edilemeyecek ölçüde büyüktür. Birinci bölümde bahsedildiği gibi görselliğin ve görselleştirmenin insan beyni için büyük etkileri vardır. "Bir resim bin sözcüğe bedeldir." denir ve bugün büyük veri çağında işletmeler çeşitli veri türlerinden, şirket içi ve bulut tabanlı kaynaklardan akan bilgi seline kapılmışken bu eski deyiş, hiç olmadığı kadar anlamlı hale gelmiştir. Görseller analizi daha kolay ve daha anlaşılır hale getirirken önemli konuları da bir bakışta görme becerisi sunar. Dahası, çoğu insan görsellere metinden çok daha iyi tepki verir. Beyne gönderilen bilgilerin %90'ı görseldir ve beyin görselleri metne kıyasla 60.000 kat daha hızlı işler[3]. Bu noktalar, bilgiyi analiz etmek ve iletmek için görselleştirmesi kullanmanın önemini güçlü bir biçimde ortaya koymaktadır.

Bu çalışmada görselleştirmenin yazılım tasarıma etkileri ele alınmıştır fakat yapılan araştırmalardan da görüldüğü üzere herhangi bir bilgi ya da veri beynimize görsel olarak girdiğinde metin olarak girdiğinden çok daha hızlı işleniyor ve kavraması çok daha kolay bir hal alıyor. Günlük hayatta karşılaşılan problemlerde, akılda tutulması gereken fakat bir türlü akılda tutulamayan verilerde, eğer öğrenci isek bilmemiz ya da ezberlememiz gereken bilgilerde, hayatın herhangi bir alanında hangi mesleğe sahip olursak olalım görselleştirmenin insan beyni için önemi bilinmeli ve karşılaşılan herhangi bir zorlukta kolaylık arayışına girildiğinde görselleştirme tekniklerin uygulamanın faydalı olacağı bilinmelidir.

6.4 Kullanılabilirlik

Geliştirilen modellemenin ve görselleştirme sisteminin kullanılabilirik açısından tartışıldığı kısımdır.

Bu çalışmada daha önce var olan bir görselleştirme sistemine karar destek sistemi eklenerek geliştirilmiştir. Yapılan çalışma incelendiğinde modelleme konusunda sağlıklı karar verip verememe kısmında eksiklik görülmüş ve bunu bir karar destek sistemi ile desteklemek düşünülmüştür. Tasarlanan karar destek sistemi görselleşitrme sistemine entegre edildiğinde kullanılabilirlik açısından bir aksaklık görülmemiş ve var olan sistemin sorunsuz olarak çalışabilmesi gözlemlenmiştir.

Yine bu çalışmada tasarlanmış olan "Apartman Dairesi Modeli" 16 yaşındaki erkek bir lise öğrencisi ve 21 yaşındaki kadın bir üniversite öğrencisi üzerinde test edilerek kullanılabilirlik konusunda herhangi bir sıkıntı yaşanmadığı gözlemlenmiştir.

7.KAYNAKÇA

[1]B. W. Boehm, Software Engineering Economics, PrenticeHall, 1981.

[2] B. Lientz, E. Swanson, and G. E. Tompkins., "Characteristics of application software maintenance", Communications of the ACM, 21(6), 1978.

[3] https://simpleshow.com/us-en/blog/visuals-v-text-what-does-the-brain-prefer/

[4] A. Endres and D. Rombach., "A Handbook of Software and Systems Engineering", Addison-Wesley, 2003.

[5] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Trans. Software Eng.,20(6), 476- 493, 1994.

[6] Panas, T., Berrigan, R. & J.C.Grundy (2003), "A 3d metaphor for software production visualization", in E. Banissi, K. Borner, C. Chen, G. Clapworthy, C. Maple, A. Lobben, C. J. Moore, J. C. Roberts, A. Ursyn & J. Zhang, eds, 'Seventh International Conference on Information Visualization, IV 2003, 16-18 2003, London,

[7] Stasko, J. T., Domingue, J., BROWN, M. H., ve Price, B. A., Eds, 1998, "Software Visualization Programming as a Multimedia Experience", The MIT Press.

[8] Richard Wettel, Michele Lanza, "Visualizing Software Systems as Cities," vissoft, 92-99 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007.

[9] Graham, H., Yang, H.Y. and Berrigan, R., "A Solar System Metaphor for 3D Visualization of Object Oriented Software Metrics", In Proc. Australasian Symposium on Information Visualization, (invis.au'04), Christchurch, New Zealand, 2004

[10] C. Lewerentz ve F. Simon., "Metrics-based 3d visualization of large object-oriented programs", In 1st Int. Workshop on Visualizing Software for Understanding and Analysis, 2002.

[11]] Holten, D.H.R., Vliegen, R., Wijk, J.J., "Visualization of software metrics using computer graphics techniques", Proceedings of the 12th Annual Conference of the Advanced School for Computing and Imaging", 135-140, Belgium, 2006.

[12] Guillaume Langelier, Houari Sahraoui, Pierre Poulin, "Visualization-based Analysis of Quality for Large-scale Software Systems", *In International Conference on Automated Software Engineering (ASE 2005)* ,7-11, 2005, Long Beach, California, USA.

[13] Termeer, M., Lange, C.F.J., Telea, A.C. & Chaudron, M.R.V., "Visual exploration of combined architectural and metric information", Proceedings 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2005), 21-26, Budapest, Hungary, 2005.

[14] M. Pinzger, H. Gall, M. Fischer, and M. Lanza, "Visualizing Multiple Evolution Metrics", InSoftVis '05: Proceedings of the 2005 ACM Symposium on Software Visualization, 67–75, New York, NY, USA, 2005. ACM Press.

[15] Lengler, R., & Eppler, M.J. (2007). Towards A Periodic Table of Visualization Methods for Management, Institute of Corporate Communication, University of Lugano, Switzerland, 6p.

[16] Forsell, C., & Johansson, Jimmy. (2010). An heuristic set for evaluation in information visualization. Proceedings of the International Conference on Advanced Visual Interfaces, Roma, Italy.

[17] Luzzardi, R.G., Freitas, C.M.D.S, Cava, R.A., Duarte, G.D. & Vasconcelos, M.H.S. (2004). An Extended Set of Ergonomic Criteria for Information Visualization. In Proc. IASTED. Int. conf. of Comp Graphics and Imaging, 236-241.

[18] Zuk, T., Schlesier, L., Neumann, P., Hancock, M.S., Carpendale, S., (2006). Heuristics for information visualization evaluation. Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization, Venice, Italy.

[19] 2 http://en.wikipedia.org/wiki/Color

[20] https://tr.wikipedia.org/wiki/Ayr%C4%B1%C5%9Ft%C4%B1rma

[21]U. Erdemir, U. Tekin, F. Buzluca, "Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi", 10/2008, s. 249- 258, Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu (YKGS08), İstanbul, 2008.

[22] https://tr.wikipedia.org/wiki/Karar_destek_sistemi

[23] https://www.metafor.com/metafor-nedir/

[24] https://www.fibiler.com/Divisions/Ehil/Mahzen/Java/TheJavaBook/txt/html/document_Qualifications.html

[25] http://www.emo.org.tr/ekler/3daba42789b6912_ek.pdf

[26]http://ceur-ws.org/Vol-1721/UYMS16_paper_50.pdf

[27]Lengler, R., & Eppler, M.J. (2007). Towards A Periodic Table of Visualization Methods for Management, Institute of Corporate Communication, University of Lugano, Switzerland, 6p.

[28]Card, S., Mackinlay J., & Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann.

[29]North C. (2005). Information Visualization, chapter in Handbook of Human Factors and Ergonomics, Third Edition. John Wiley & Sons, New York, 1222-1246

[30] 3. Luzzardi, R.G., Freitas, C.M.D.S, Cava, R.A., Duarte, G.D. & Vasconcelos, M.H.S. (2004). An Extended Set of Ergonomic Criteria for Information Visualization. In Proc. IASTED. Int. conf. of Comp Graphics and Imaging, 236-241.

[31]Luzzardi, R.G., Freitas, C.M.D.S, Cava, R.A., Duarte, G.D. & Vasconcelos, M.H.S. (2004). An Extended Set of Ergonomic Criteria for Information Visualization. In Proc. IASTED. Int. conf. of Comp Graphics and Imaging, 236-241.

[32] 8. Scapin, D.L., & Bastien, J.M.C. (1997). Ergonomic criteria for evaluating the ergonomic quality of interactive system. Behaviour and Information Technology, 16(4-5), 220-231.

[33]Ware, C. (2004). Information Visualization: Perception for Design, Second Edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[34]Zuk, T., Schlesier, L., Neumann, P., Hancock, M.S., Carpendale, S., (2006). Heuristics for information visualization evaluation. Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization, Venice, Italy.

[35] Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.). Usability Inspection Methods. John Wiley & Sons, NY, USA, 25-61.

[36]Shneiderman, B. (1998). Designing the User Interface: Strategies for Effective HumanComputer Interaction, Third Edition. Addison-Wesley, Reading, MA.

[37]Bresciani, S., & Eppler, M. (2015). Extending Tam to Information Visualization A Framework for Evaluation. Electronic Journal of Information System Evaluation, 18(1), 46-58

[38]Forsell, C., & Johansson, Jimmy. (2010). An heuristic set for evaluation in information visualization. Proceedings of the International Conference on Advanced Visual Interfaces, Roma, Italy

[39]Nielsen, J. (1993). Usability Engineering. Academic Press, Boston

[40] Cawthon, N., & Moere, A.V. (2007). The effect of aesthetic on the usability of data visualization. In IEEE Int. Conf. Info Vis (IV',07), Zurich, Switzerland, 637-648.

[41] Moere, A.V., Tomitsch, M., Wimmer, C., Christoph, B., & Grechenig T. (2012). Evaluating the effect of style in information

visualization. IEEE Trans. Vis. Comput. Graph., 18(12), 2739-2748

[42] https://docplayer.biz.tr/26060972-Bilgi-gorsellestirme-tekniklerinin-yazilim-kullanilabilirligi-acisindan-degerlendirilmesi.html