T.C.

ULUDAĞ ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ ANABİLİM DALI

PROGRAMLAMA ÖĞRETİMİNDE KULLANILABİLECEK YAZILIMLARA İLİŞKİN ÖĞRETMEN GÖRÜŞLERİ

YÜKSEK LİSANS TEZİ

Salih BALTALI

BURSA

2016

T.C.

ULUDAĞ ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ ANABİLİM DALI

PROGRAMLAMA ÖĞRETİMİNDE KULLANILABİLECEK YAZILIMLARA İLİŞKİN ÖĞRETMEN GÖRÜŞLERİ

YÜKSEK LİSANS TEZİ

Salih BALTALI

Danışman

Doç. Dr. Adem Uzun

BURSA

2016

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim.

Salih BALTALI

26/04/2016

YÖNERGEYE UYGUNLUK ONAYI

"Programlama Öğretiminde Kullanılabilecek Yazılımlara İlişkin Öğretmen Görüşleri" adlı Yüksek Lisans tezi, Uludağ Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan	Danışman
Salih BALTALI	Doç. Dr. Adem UZUN

Bilgisayar ve Öğretim Teknolojileri Eğitimi ABD Başkanı

Prof. Dr. Aysan ŞENTÜRK

T.C.

ULUDAĞ ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar ve Öğretim Teknolojileri Eğitimi Ana Bilim Dalı'nda 801390005 numaralı Salih BALTALI' nın hazırladığı "Programlama Öğretiminde Kullanılabilecek Yazılımlara İlişkin Öğretmen Görüşleri" konulu Yüksek Lisans çalışması ile ilgili tez savunma sınavı, 11/03/2016 günü 13:30 -14:30 saatleri arasında yapılmış, sorulan sorulara alınan cevaplar sonunda adayın tezinin/çalışmasının **(başarılı / başarısız)** olduğuna **(oy birliği / oy çokluğu)** ile karar verilmiştir.

Üye

(Tez Danışmanı ve Sınav Komisyonu Başkanı)

Doç. Dr. Adem Uzun

Uludağ Üniversitesi

Üye

Doç. Dr. Rüçhan Özkılıç

Uludağ Üniversitesi

Üye

Yrd. Doç. Dr. Funda Dağ

Kocaeli Üniversitesi

ÖNSÖZ

Bilgisayar bilimleri her zaman önemli bir bilim dalı olagelmiştir. Bilgisayar bilimleri denildiğinde akla sadece programlama ve sayfalarca yazılmış kodlar gelmemelidir. Bunlar sadece işin dışarından görünen tarafı olup diğer taraftan mantıksal (algoritmik) düşüncenin nasıl kullanılacağını, karışık problemleri çözme yeteneği ve kritik düşünme becerisinin kazandırılmasını sağlar. Bu kabiliyetlerin erken yaşlarda öğrencilere kazandırılması gerekmektedir. Bugün pek çok ülkede yapılan programlama öğretiminin sağlanabilmesi için bir takım görselleştirme araçları (yazılım) üretilmiştir. Bu araçlar öğrenciye soyut programlama kavramlarını öğretmek, algoritmik düşünme becerisini kazandırma iddiasında olan araçlardır. Ancak programlama öğretiminde kullanılacak olan bu araçların tasarım ve kullanılabilirlik açısından hitap ettiği öğrenci kesimine ne kadar uygun olduğu üzerinde durulması gereken bir konudur.

Araştırmada emeği geçen başta danışmanım Doç. Dr. Adem Uzun'a, Yrd. Doç. Dr. Erhan Şengel'e, Yrd. Doç. Dr. Semiral Öncü'ye, yüksek lisans arkadaşım Beyhan Çıtak'a ve çalışmanın oluşmasında katkılarını esirgemeyen Bursa Bilişim Teknolojileri öğretmenlerine teşekkür ederim. Ayrıca manevi desteğini hiç esirgemeyen kıymetli eşim Emine Baltalı' ya da teşekkürü bir borç bilirim.

Salih Baltalı

Özet

Yazar	: Salih BALTALI
Üniversite	: Uludağ Üniversitesi
Ana Bilim Dalı	: Bilgisayar ve Öğretim Teknolojileri Eğitimi
Bilim Dalı	: Bilgisayar ve Öğretim Teknolojileri Öğretmenliği
Tezin Niteliği	: Yüksek Lisans Tezi
Sayfa Sayısı	: XVII + 185
Mezuniyet Tarihi	: 27 / 05 / 2016
Tez	:Programlama Öğretiminde Kullanılabilecek Yazılımlara İlişkin Öğretmen Görüşleri
Danışmanı	: Doç. Dr. Adem Uzun

PROGRAMLAMA ÖĞRETİMİNDE KULLANILABİLECEK YAZILIMLARA İLİŞKİN ÖĞRETMEN GÖRÜŞLERİ

Bu çalışmanın amacı ilköğretim seviyesinde programlama öğretiminde kullanılabilecek bazı yazılımların BT öğretmenlerinin görüşlerini alıp kullanılabilirlik ve tasarım açısından uygun olup olmadığını incelemektir. Çalışmada Scratch 1.4, Microsoft Small Basic 1.2, Microsoft Kodu Game Lab 1.4.64 ve Robomind 6.01 yazılımları incelenmiştir. Yazılımların incelemesine Bursa ilinde ortaokulda görev yapan 92 Bilişim Teknolojileri (BT) öğretmeni katılmıştır. Rastgele paylaştırılan bu yazılımların arayüz tasarımı ve kullanılabilirlik incelemesinde Jacob Nielsen'in (2015a) sezgisel (heuristic) rehberi esas alınmıştır. 2005 yılında yayınlanan bu rehber tekrar düzenlenerek elektronik bir form halinde öğretmenlere iletilmiştir. Araştırma nicel bir yapıya sahip olup tekil tarama modeline göre; 12 başlık altında yer alan sorulara katılımcılar Evet, Hayır

ve Uygulaması Yok cevapları vermişlerdir. Elde edilen verilerin frekans, ortalama ve yüzdeleri hesaplanıp çalışma gurubunun araştırma konusuna göre fikri yönü araştırılmıştır.

Yapılan araştırma sonucunda programlama öğretiminde görsel araç kullanımının ilköğretim öğrencileri için pozitif etkiler bıraktığı ve bu araçların genel itibariyle ortaokul kademelerine uygun olduğu görülmüştür. Ancak Small Basic 1.2 yazılımının ortaokulda kullanılabileceği gibi içerdiği kod tabanlı yazım düzeniyle profesyonel dillere daha yakın durduğundan lise kademesinde de yararlanılabileceği ortaya çıkmıştır. Diğer taraftan, BT öğretmenlerine göre yazılım hazırlanırken %8 kullanılabilirlik, %4 tasarım, %80 her ikisine dikkat edilmesi gerektiği ortaya çıkmıştır. Görselleştirme yazılımlarının genel itibariyle kullanılabilir ve tasarımlarının da iyi olduğu, bunlar arasından Robomind 6.01 yazılımının en kullanılabilir ve tasarım bakımından en az soruna sahip olduğu sonucuna ulaşılmıştır.

*Anahtar sözcükler***:** görselleştirme yazılımı, kullanılabilirlik, programlama, programlama öğretimi, yazılım tasarımı

Abstract

Author	: Salih BALTALI
University	: Uludağ University
Field	: Computer and Instructional Technologies Education
Branch	: Computer and Instructional Technologies Teacher
Degree Awarded	: Master Thesis
Page Number	: XVII + 185
Degree Date	: 27 / 05 / 2016
Thesis	:Teachers' Views on Software That Can Be Used in Teaching Programming
Supervisor	: Doç. Dr. Adem Uzun

TEACHERS' VIEWS ON SOFTWARE THAT CAN BE USED IN TEACHING PROGRAMMING

The purpose of this study is to determine whether some software that can be used to teach programming in primary school level is appropriate in terms of usability and design subsequent to taking IT teachers' views. Scratch 1.4, Microsoft Small Basic 1.2, Microsoft Kodu Game Lab 1.4.64 and Robomind 6.01 software programs were analyzed in the study. 92 Information Technologies (IT) teachers working at a primary school in the province of Bursa participated in the analysis of the software. In the interface design and usability evaluation of these software programs that were equally shared between the teachers, Jacob Nielsen's (2015a) heuristic guideline was predicated on. First published in 2005, this guideline was delivered to the teachers in an electronic form by being reedited. The research was in a quantitative design. According to

the single screening model, the survey consisted of 12 titles of which the participants answered the questions as “Yes”, “No” or “Not Available”. The opinions of the teachers as the group were investigated by obtaining statistics involving frequencies, means and percentages of the data gathered.

As a result of the research, it was found out that the use of visual tools for primary school students in teaching programming left positive impressions and generally those programs were appropriate for middle school level. Nevertheless, the research indicated that Small Basic software could be used not only in middle schools, but also could be utilized in high school level as it is comprised of text based programming layout and interface. On the other hand, according to the IT teachers, while a software of programming was being written, 8% of usability, 4% of design and 80% of both aspects should be taken into consideration. The visualization software programs are generally useable and have a good design; among other software programs, Robomind 6.01 software was found out to be the best useable and to have the least problems in terms of design.

Keywords: design of software, programming, teaching programming, visualization software, usability

İçindekiler

Sayfa No

Tablolar Listesi

Tablo *Sayfa*

Şekiller Listesi

Kısaltmalar Listesi

BT: Bilişim Teknolojileri

EBA: Eğitim Bilişim Ağı

ISO: Uluslararası Standartlaşma Teşkilatı

İBE: İnsan Bilgisayar Etkileşimi

MEB: Milli Eğitim Bakanlığı

TDK: Türk Dil Kurumu

1. Bölüm

Giriş

Bilgi, her yüzyılda farklı bir şekilde kullanılmış; insanoğlu, onu günlük hayatında farklı formlara sokarak bir düşünce akımı, kuram ya da bir aygıt haline getirmiştir. Örneğin, elektriğin keşfiyle birlikte peşi sıra bulunan onlarca elektronik devre elemanı sayesinde, zaman içerisinde telefon, hesap makinesi, bilgisayar gibi hayatımızda farklı ihtiyaçlarımızı karşılayan araçlara/aygıtlara dönüşmüştür. Bilgisayarlar sizin ona verdiğiniz komutları, talimatları yerine getiren elektronik ve mekanik cihazlardır. Elektronik bir cihazı yönetebilmek için ona anlayabileceği komutların verilmesi gerekmektedir. İşte tam bu noktada bilgisayar teknolojisi içerisinde ayrı bir yeri olan yazılım dünyası karşımıza çıkar. Yazılım, bilgisayarlar ile insanoğlu arasındaki iletişimi sağlayan bir arayüz ya da iletişim dilidir denilebilir. İlgi ve ihtiyaçlarımızı bilgisayarların anlayabileceği şekilde komutlar halinde ona girdiğimizde isteklerimizi yerine getirecektir. Bu komutların oluşturulması ve bir araya getirilmesi işlemine programlama denilmektedir.

1980'lerde sanayi toplumundan bilgi toplumuna geçiş yapıldığında temel bilgisayar bilgi ve becerilerinin de artık okuma yazma becerisinden sonra elde edilmesi gereken bir yeterlik olduğu tartışılmaya başlanmıştır (Drucker, 2000). Çoğunlukla bilginin değerli olduğu ve sadece ona erişmekle yetinilen ve bunun yeterli bulunduğu 1980-1990'lı yıllarda programlamanın önemi yavaş yavaş kavranmaya başlanıp 2000'li yıllarda bilgiye erişimden bilgi ve içerik üretimine geçiş yapılmıştır. Artık farklı ihtiyaçları karşılayabilecek yazılımlar programlama sayesinde üretilip insanlığın hizmetine sunulmuştur. Bilgisayarların eğitim, sağlık, ulaşım, haberleşme gibi birçok sektörde varlık göstermesi ve yayılması programlamanın cazibesini hiçbir zaman yitirmemesini sağlamıştır.

Programlamanın önemini bilen ve gelecekte kendilerine neler kazandırabileceğini öngören ülkelerde, eğitim politikaları içerisine dâhil edilip, uzun yıllardan bu yana çeşitli öğretim kademelerinde programlama eğitimi verilmiş, programlama öğretiminin önemli olduğu birçok kaynakta vurgulanmış ve konuyla ilgili çalışmalar yapılmıştır. Buna göre, (Tucker et al., 2003) araştırmalarında Avrupa, Rusya, Asya, Güney Afrika, Yeni Zelanda ve Avustralya'da bilgisayar bilimleri dersinin K–12 öğretim programına eklendiğini belirtmektedirler. İsrail ve Kanada'da ise programlama eğitimi ile ilgili dersler liselerde verilmektedir.

Dünyada programlama dersleri, mesleki yeterlilik için gerekli olmanın yanında bilgisayar okuryazarlığının önemli bir parçası olmaları nedenleriyle yaygınlaşmaktadırlar. Zira bu tür derslerin öğrencilerin analitik düşünme ve problem çözme becerilerinin gelişmesi açısından faydalı olduğu bilinmektedir (Sleeman, Putnam, Baxter & Kuspa, 1984). Bu nedenlerden ötürü dünyada birçok ülkede programlamaya giriş dersleri ilk ve ortaöğretim öğretim programlarına eklenmiştir (Tucker et al., 2003).

1.1. Problem Durumu

Programlama öğretiminin ilköğretim düzeyinde verilmeye başlanmasıyla çocuklara programlama öğretiminin nasıl verileceği sorunu ortaya çıkmıştır. Eğitim kurumlarında programlama öğretimi genelde teorik bir yöntemle verilmektedir. Teorik yöntemin öğretimde hem sıkıcı hem de çok etkili olmadığı bilinmektedir (Arabacıoğlu, Bülbül ve Filiz, 2007). Programlama öğretimi kavramlarından olan koşul, döngü, şartlı döngü gibi temel programlama kavramlarının teorik olarak aktarılması öğretimi zorlaştırabileceği düşünülmektedir. Soyut olarak geçen bu kavramlar zaten zihinsel süreçleri bakımından somut evreden soyut evreye geçtikleri ortaokul yaşlarında öğrenci zihninde tam olarak oturamamakta; amaç, işlev ve sonuçları tam anlamıyla anlaşılamamaktadır. Bu problemi aşmak ve programlama kavramlarının

anlaşılabilmesi için uzun yıllardan beri kullanılan öğretim yazılımları ve görselleştirme araçları bulunmaktadır. Bazı ülkeler öğretim programlarının içerisinde bu yazılımları kullanarak programlama öğretimi yapılmasını yönünde uygulamalar geliştirmişlerdir. Ancak bahsi geçen yazılım ve görselleştirme araçlarının belirli yaş ve sınıflara uygulanmasında öğrencilerin mevcut zihinsel algı durumları göz önüne alınırsa yazılımların kullanılabilirlik ve tasarımları açısından sorunlar oluşturacağı açıktır. Tasarım bakımından zayıf, o yaş veya sınıfa hitap etmeyen, kullanılabilirliği düşük olan bir programlama öğretimi aracının vereceği bir şey olamayacağı gibi karmaşıklığa ve yanlış öğrenmeye sebep olma ihtimali de vardır. Bu yüzden öğretimde kullanılacak yazılımların o dersi verecek öğretmen tarafından doğru seçilmesi veya öğretmen görüşlerinin alınması gerekmektedir.

1.2. Amaç

Bu araştırmanın amacı, programlama öğretiminde kullanılabilecek Scratch 1.4, Small Basic 1.2, Microsoft Kodu Game Lab 1.4.64 ve Robomind 6.0.1 yazılımlarını Nielsen'in (2015a) sezgisel kontrol aracına göre BT öğretmenlerinin görüşlerini kullanılabilirlik ve tasarım açısından analiz etmektir.

1.3. Önem

Kelleher, Pausch ve Kiesler (2007), programlama eğitiminde görselliğin önemini vurgulamak amacı ile gerçekleştirdikleri araştırma sonucunda, görsel özelliklerin öğrenci başarı ve motivasyonunda önemli etkisi olduğunu belirlemişlerdir. Konu ile ilgili yapılan diğer araştırmalar, yazılımlarda grafik ve animasyon kullanımının öğrencilerin derslere olan ilgisini arttırmada daha etkili olduklarını göstermektedir (Bishop-Clark, Courte & Howard, 2007; Lin & Zhang, 2003). Tüm bu nedenlerden ötürü, görsellik ve canlandırma içeren yardımcı araçlarla yapılacak uygulamaların uygun yöntemlerle derslere dâhil edilmesi durumunda, ilköğretim

öğrencilerinin programlamaya ilgi duymalarını ve sevmelerini sağlayabileceği söylenebilir. Araştırmalar, yazılımda kullanılan görselliğin ve etkileşimin öğrenci başarısı üzerindeki etkisini vurgulamaktadır (Arabacıoğlu ve diğerleri, 2007; Gültekin, 2006). Lahtinen, Ahoniemi ve Salo (2007), araştırma sonuçlarına göre problemlerin karmaşıklığı arttıkça görselleştirme araçlarına daha çok ihtiyaç duyulmaktadır. Yapılan araştırmalar üniversite düzeyinde gerçekleştirilmiş olup, ilkokul ve ortaokul seviyesine uygun görselleştirme aracının belirlenmesinde yetersiz kalmaktadır. Bu durumda ilkokul ve ortaokul seviyesine programlamanın temelleri öğretilirken, hangi görselleştirme araçlarını kullanmanın ve bunların kullanılabilirliği ve tasarımı bakımından daha uygun olacağının araştırılmasında yarar vardır. Ayrıca programlama öğretiminde yapılan araştırmalara bakıldığında, genellikle kullanılan görselleştirme araçlarının programlama başarısına, motivasyona etkileri gibi konular işlenmiştir. Görselleştirme araçlarının programlama öğretiminde kullanılabilirliği ve tasarımı bakımından değerlendirmesine dair bakılan kaynaklar arasında rastlanamamıştır. Bu da yapılan bu araştırmayı önemli kılan bir başka etkendir.

1.4. Araştırma Soruları

Araştırma kapsamında aşağıdaki sorulara cevaplar aranacaktır:

1. Bilişim Teknolojileri (BT) öğretmenleri programlama öğretiminde kullanılan görselleştirme araçlarının kullanımını nasıl değerlendirmektedir?
2. BT öğretmenleri programlama öğretiminde kullandıkları görselleştirme araçlarında kullanılabilirlik veya tasarım kavramlarından hangisine daha çok önem vermektedirler?
3. Programlama öğretiminde ilkokul, ortaokul ve lise kademelerinde hangi görselleştirme araçları kullanılmalıdır?
4. Tasarım ve kullanılabilirlik açısından en az probleme sahip yazılım hangisidir?

5. Kullanılan görselleştirme yazılımlarının barındırdıkları tasarım ve kullanılabilirlik problemleri nelerdir?

1.5. Varsayımlar

Araştırmada kabul edilen varsayımlar aşağıdaki gibidir:

- Seçilen örneklemin evreni temsil ettiği varsayılmaktadır.
- BT öğretmenlerinin kendilerine verilen görselleştirme aracını bilgisayarına indirip, kurup bireysel olarak kullandığı ve buna göre kontrol aracındaki soruları yanıtladığı varsayılmıştır.
- BT öğretmenlerinin kendilerine verilen sorulara içtenlikle ve yansız cevaplar verdikleri varsayılmıştır.

1.6. Sınırlılıklar

- Araştırma, 2014-2015 eğitim öğretim yılında Bursa'da görev yapan 92 BT öğretmeni ile sınırlıdır.
- Araştırma, son dönemin popüler olan programlama öğretimi araçlarından Scratch 1.4, Small Basic 1.2, Microsoft Kodu Game Lab 1.4.64 ve Robomind 6.01 yazılımları ile sınırlıdır.

1.7. Tanımlar

Program Görselleştirme: Bu tip görselleştirme aracı, bir programın ve onun verilerinin nasıl çalıştığını grafiksel olarak sunmak için kullanılır (Bergin & Martinez,1996).

Algoritma Animasyonları: Temel algoritmaların hangi işlemleri tamamladığını görseller yardımıyla göstermek amacıyla kullanılır (Bergin & Martinez,1996).

Veri Görselleştirme: Bu tip görselleştirme araçları ise program kodu içerisinde veri tiplerinin neler olduğu ve hangi işlemlerden sonra son durumlarının ne olduğu hakkında bilgi verir.

Görsel programlama araçları: Programları bir ara yüz yardımıyla, sürükle–bırak biçiminde yapılandırmayı sağlayan araçlardır (Cooper et al., 2006)..

Program çıktılarını görselleştiren araçlar: Oluşturulan programlar yürütüldüğünde çoklu ortam kullanarak dönüt veren araçlardır (Cooper et al., 2006).

Sıraya dizilmiş programlama dili araçları: Programlama dilini öğrenci seviyesine göre uygun biçimde aşamalı olarak öğreten araçlardır (Cooper et al., 2006).

Algoritmayı hikayeleştiren araç: Programlamayı bir animasyon karakter yardımıyla hikayeleştirerek öğretmeyi amaçlayan görselleştirme aracıdır (Cooper et al., 2006).

Akış Şeması Modelli araç: Program parçalarını işlem sırasını gösterecek biçimde birbirine bağlayarak programları yapılandırmayı sağlayan görselleştirme aracıdır (Cooper et al., 2006).

2. Bölüm

Literatür

Bu bölümde araştırmanın başlığını oluşturan insan bilgisayar etkileşimi, kullanılabilirlik, tasarım ve programlama öğretiminin önemi, karşılaşılan zorluklar, öğretimde içine düşen yanılgılar, öğretimde kullanılan eski ve yeni yazılımlar ve öğretimin nasıl değerlendirilmesi gibi konular alt başlıkları ile detaylıca incelenecektir.

2.1. İnsan Bilgisayar Etkileşimi

2.1.1. İnsan bilgisayar etkileşimi nedir? İnsan yaşamında olmazsa olmaz hale gelen bilgisayar birçok insan tarafından kabul görmüştür. Her insanın verimli bir şekilde kullanabilmesi için kullanıcı etkileşimine olanak sağlayan arayüzün doğru tasarlanması gerekmektedir. Olson ve Olson'a (2003) göre yazılım ve donanıma sahip sistemlerin tasarımlarında arayüzler, kullanıcının sistemle etkileşim sağlaması ve sistemi öğrenmesi için harcanan çabayı etkileyen bir unsurdur. Bu bağlamda İnsan-Bilgisayar Etkileşimi teknolojik arayüzlerle kullanıcının etkileşimini inceleyen bir çalışma alanıdır. Başlangıçta bilgisayar sistemleri ve yazılımların tasarımlarını araştırmak kullanıcı gözüyle yeteri kadar önemli görülmemiş, ancak zamanla bu anlayış değişmiştir. Bu nedenle ilk olarak Bilgisayar İnsan Etkileşimi (Computer-Human Interaction) adlandırılan bu alan etkileşimin insan/kullanıcı boyutunun da olduğunun farkedilmesiyle İnsan-Bilgisayar Etkileşimi (İBE) olarak değişime uğramıştır (Hewett et al, 2004). İBE alanındaki çalışmaların temel amacı Olson ve Olson'a göre (2003), kullanıcıların belirli kullanım ortamlarında ve belirli görevleri yerine getirirken ürün arayüzleri ile aralarındaki etkileşimi sağlayıp kullanıcıların ihtiyaçlarına karşılık veren, kullanılabilirliği yüksek ve kullanıcı dostu yazılım ve donanımlar geliştirmektir. Hartson (1998) ise bu amacı etkileşimli (interactive)

sistemlerin kullanıcının etkinliğini ve memnuniyetini artırmak için sistemin tasarımı, geliştirilmesi ve değerlendirilmesi olarak açıklamıştır. Şengel ve Özdemir de (2012) teknolojiyi insanın daha doğal olarak kullanılabilecek hale getirilmesi çabası olarak ifade etmişlerdir.

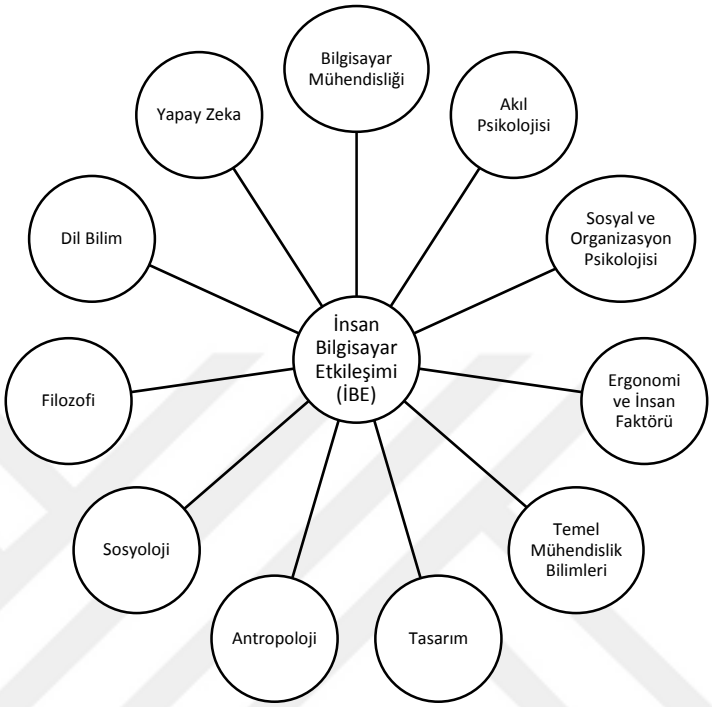
İBE alanı, bilgi teknolojilerine dayalı sistemlerin daha kullanılabilir hale getirilmesi ve kullanıcının ihtiyaçlarına karşılık veren sistemler üretilmesi konusu ile ilgilenen içinde birçok disiplinli barındıran bir alandır (Şengel ve Özdemir, 2012). İBE, hem teknoloji ve hem de insanı içinde barındırmaktadır. Teknoloji yönüyle yazılım, donanım ve fiziksel tasarım gibi alanlar, insan/kullanıcı bakımından ise psikoloji, iletişim, sosyoloji ve organizasyonel bilimler iyi bilinmesi gerekmektedir (Çağıltay, 2011). Bu bakımdan İBE, psikoloji, insan davranışı, bilgisayar teknolojileri, bilişsel bilimler ve yazılım mühendisliği alanlarının yanı sıra grafik, ergonomi, endüstriyel tasarım, sosyoloji, antropoloji ve eğitim bilimleri gibi farklı disiplinlerle de ilişkilidir (Preece, 1994; Shneiderman, 1998).

İnsan Bilgisayar Etkileşimi tanımlarının sahip olduğu genel özellikleri aşağıdaki gibi özetleyebiliriz:

- **Disiplinler arası çalışması:** Birçok disiplinle çalışarak yaşanılan problemlerin çözümler üretmeye çalışır.
- **Kullanılabilirlik:** Bu alanın en temel çalışma konusudur. Teknolojiyi daha kolay kullanılabilir kılma ve etkileşimi artırma gibi konular üzerinde çalışır.
- **Tasarım:** Kullanımı ve fonksiyonelliği yüksek ürünlerin nasıl tasarlandığını araştırır.
- **Etki:** Teknolojinin insan yaşamı üzerine etkileri konusunu araştırır (Çağıltay, 2011).

Şekil 2.1

İnsan Bilgisayar Etkileşiminin Bağlı Olduğu Temel Disiplinler (Preece, 1994)



Çağıltay (2011) çalışmasında İBE kavramının ilk olarak 1945'te Vannevar Bush tarafından ortaya atılan teorik analog bilgisayar fikrinden ortaya çıktığını belirtir. Bush'un (1945) "As We May Think- Düşündüğümüz Gibi" adlı makalesinde Memex isimli bir makineden bahseder. Makalede, bu makinenin insanın bilişsel sistemini nasıl desteklediği ve sadece insan bilgisayar değil aynı zamanda bu makine ile yapılan insan-insan arasındaki etkileşimden de söz edilmektedir. Çalışmada araştırmacılar hayali olarak yer alıp, tarihte çok ünlü olan Türk ok ve yayları konusunda araştırma yapmaktadırlar. Bu araştırmacılar, Memex'in zengin kütüphanesiyle gerekli bilgiye ulaşıp, elde ettikleri bu bilgiyi birbirleri ile paylaşmakta ve daha sonra kullanmak üzere yine Memex'de depolamaktadırlar.

İngiltere'de Brian Schackel'in, 1959 yılında Design dergisinde yazmış olduğu "Ergonomics for a Computer- Bilgisayar Ergonomisi" adlı makale İBE konusunda yayınlanmış

ilk makale olarak gösterilebilir (Schackel, 1997). *Human Factors (1959)* ve *International Journal Of Man-Machine Studies (1969)* adlı dergiler ise bu alanda yayınlanmış ilk bilimsel dergilerdir. İnsanların bilgisayarlar ile olan etkileşimlerinin, insanlar arasındaki etkileşim gibi olmasının gerektiğini ileri süren Eric Licklider'in 1960 yılında ortaya attığı "*Man-Computer Symbiosis*– İnsan Bilgisayar Bütünleşmesi" fikri ise hala gerçekleştirilememiştir (Schackel, 1997). Benzer bir diğer yaklaşım ise daha sonra Lucy Suchman tarafından ortaya atılmıştır (Suchman, 1987). Suchman, kullanılabilir bilgisayar sistemleri tasarlamak yerine, insanları anlayan sistemler geliştirmenin gerekli olduğunu düşünmektedir.

İBE'nin daha sonraki tarihsel gelişim süreci ise şu şekilde devam etmiştir; 1969 yılında "International Journal of Man-Machine Studies" adlı bilimsel yayın hayatına başlamıştır. 1970'ler de daha sonra sıkça duyacağımız "userfriendly- kullanıcı dostu" kavramı ortaya atılmıştır. 1971'de Gerald M. Weinberg, "ThePsychology of Computer Programming", 1980 yılında Ben Shneiderman, "Software Psychology: Human Factors in Computerand Information Systems" adlı kitaplarını yazmışlardır (Suchman, 1987). 1980'lerden bu tarafa İBE alanındaki araştırmalar hızla artmış ve konuyla ilgili çok sayıda uluslararası bilimsel dergi ve kitap yayımlanmıştır (*Behaviourand Information Technology, Interactions, International Journal of Human- Computer Interaction, ACM Transactions on Computer-Human Interaction* gibi). 1990'lar da alandaki çalışmalara yönelik birçok profesyonel dernek ve araştırma merkezleri kurulmuş, konferanslar düzenlenmiştir. İBE çalışmaları birçok disiplinin eğitim programları arasında yer almaya başlamıştır. 1990'ların sonuna doğru, Carnegie Mellon Üniversitesi ve Indiana Üniversitesi gibi birçok üniversitede İBE alanında akademik programlar başlatılmıştır (Çağıltay, 2011).

2.1.2. İBE Neden Önemlidir? İBE, bilgisayarların ve teknolojik araç-gereçlerin günlük yaşama girmesiyle önemi daha da artmıştır. Bir taraftan nüfusun artması, birçok teknolojik araçların ortaya çıkması ve bunların sahip olduğu özelliklerin çoğalması, her kesimden insana hitap etmesi, diğer taraftan akademik camiada insan davranışı ve zihinsel süreçler hakkında doğru bilinen yanlışların terk edilmesi İBE'nin önemini artırmıştır. Çağıltay (2011), İBE'nin önemini;

- Kullanıcı nüfusunun artması ve çeşitlenmesi;
- Organizasyonların bilişim sistemlerine bağımlılığı;
- Teknolojinin kritik uygulama alanları;
- Donanım masraflarının düşmesi ve insan masrafının artması;
- Üretkenlik ikilemi;
- İnsanın bilişsel (zihinsel) durumu ve davranışlarını anlama çabaları olarak sıralamıştır.

İBE alanında yapılacak çalışmaların sağladığı en önemli fayda, bu konudaki bilinmeyenleri araştırarak, daha rahat kullanımı olan bilişim sistemleri geliştirmek, teknoloji üreticilerini bu konu hakkında bilgilendirmek, üzerine çalışacak araştırmacılar yetiştirmek ve konu ile ilgili ulusal ve evrensel bilimsel birikime katkıda bulunmaktır (Çağıltay, 2011).

2.1.3. İBE ana bileşenleri. İnsan Bilgisayar Etkileşiminin dört ana bileşeni vardır:

1. **Kullanıcı** (user)**:** Teknolojik ürünler insana hizmet için yapıldığından insanın, ürün kullanımından mutluluk duyması ve verimli çalışması adına her türlü geliştirme bu çerçevede düşünülebilir. Kullanıcı burada tek bir kişi ya da bir grup olabilir.
2. **Araç / Arayüz** (tool)**:** Kullanıcının gerçekleştirmek istediği görevi yerine getirmek için kullandığı her türlü teknolojik üründür. Örnek olarak bilgisayar, telefon, web sitesi gibi.

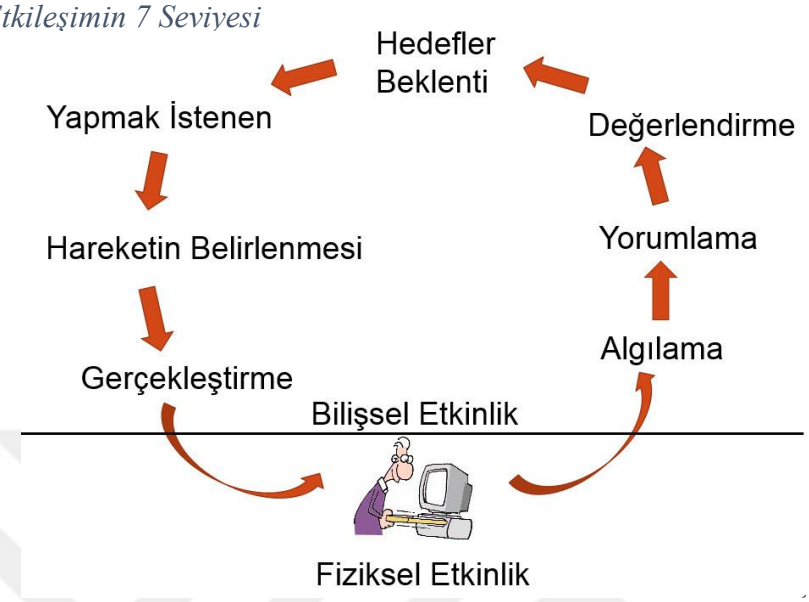
3. **Görev** (task)**:** Kullanıcının elindeki teknolojik ürün ile gerçekleştirdiği iştir. İBE'nin öncelikli işi görevin kolayca gerçekleştirilmesini sağlamak ve iyileştirmektir.
4. **Bağlam** (context)**:** Kullanıcının bir görevi teknolojik bir ürün ile gerçekleştirirken içinde bulunduğu ortamdır. Bu bir ofis, okul, ev, sokak olabilir (Çağıltay, 2011).

2.1.4. İBE'nin felsefi boyutu. Araştırmacılar İnsan Bilgisayar Etkileşimine bakışı iki ana yaklaşımla ele almışlardır. Birinci yaklaşım, etkileşimde kullanıcının bilgiyi nasıl işlediği ve nasıl davrandığı esasına dayanmaktadır. Bu yaklaşımın önde gelen araştırmacıları arasında Donald Norman ve Ben Shneiderman sayılabilir. Norman(1988), bir bilgisayar kullanıcısının bilgisayar ile etkileşim sürecini, belirli işlem seviyelerine bölerek modellemeye çalışmaktadır. Shneiderman(1998) ise deneysel yaklaşımlar ile elde ettiği bulgular doğrultusunda, bilgisayar arayüzlerinin nasıl daha iyi tasarlanabileceği konusunda önerilerde bulunmaktadır (Çağıltay, 2011). İkinci yaklaşıma göre, etkileşim sürecinin iletişim ve bağlam özellikleri öne çıkarılmaktadır. Yani Norman'ın bakış açısından farklı olarak bilgisayar kullanıcıyı anlamalı, ne yapmak istediğini sezip ona göre davranmalıdır. Lucy Suchman'ın (1987) bakış açısı daha anlaşılabilir bir durumu tanımlamakla beraber, günümüz teknolojisi henüz bu yaklaşımı gerçekleştirecek seviyeye ulaşamamıştır (Çağıltay, 2011).

Norman (1988), yaklaşımını modelleyerek iki etkinlik üzerine kurmuştur. Bunlar birisi fiziksel, ikincisi ise bilişsel etkinliktir. Fiziksel etkinlik olarak fareyi tıklamak, tuşlara basmak ya da oyun çubuğunu hareket ettirmek gibi örnekler sayılabilir. Etkileşimin gözle gözlenemeyen boyutu olan bilişsel etkinliği Norman 7 seviyeye ayırmıştır (Çağıltay, 2011).

Şekil 2.2

Norman'a Göre Etkileşimin 7 Seviyesi



Algılama: Etkileşim sürecinin ilk basamağıdır. Örneğin ekrandaki hata mesajını görsel algı sistemi ile algılamaktadır.

Yorumlama: Görsel olarak algılanan bu bilginin bu adımda yorumlanması gerekmektedir.

Değerlendirme: Ekrandaki hata mesajını, önem sırasına göre "önemli ya da değil" gibi bir değerlendirme sürecidir.

Hedefler/Beklenti: Yapılacak işe göre amaç veya hedefin ne olduğunun belirlendiği süreçtir.

Yapmak istenen: Zihinde oluşan ilk adım, bir şey yapmaya ya da yapmamaya karar verilmesi noktasıdır.

Hareketin belirlenmesi: Zihinden "tepki verilecek" yanıt geldiyse ardından ilgili hareket belirlenir. Harekete örnek olarak farenin sol tuşuna tıklamak veya klavyede belli bir tuşa basmak gösterilebilir.

Gerçekleştirme: Son adım olan bu aşamada vücudun ilgili kaslarına verilen emir ile bilişsel etkinlik fiziksel bir tepki ile sonlandırılır. Bu süreç döngüsel olarak sürekli devam eder (Çağıltay, 2011).

Donald Norman hazırlamış olduğu bu modelde iki önemli kavram üzerinde durmuştur. Bu kavramlar; hedefe varana kadar geçen süreç "Değerlendirme", hedefin artık fiziksel bir eyleme dönüştüğü süreç olan "Gerçekleştirme"dir. Ürün kullanılabilirliğinde sorunlar, Norman'ın tabiriyle "Değerlendirme Körfezi" ve "Gerçekleştirme Körfezi" nedeniyle oluşmaktadır. Nasıl ki bir körfezde iki uç arasındaki mesafe arttıkça karşıya geçiş o kadar zor oluyorsa, insan bilgisayar etkileşiminde de benzer bir durum vardır. Ekrana çıkan bir mesajı kullanıcı anlayamıyorsa "Değerlendirme Körfezi" açıklığı büyüktür demektir. Aynı şekilde kullanıcı mesajı anlayıp ne yapacağını kestiremiyorsa bu kez de "Gerçekleştirme Körfezi" açıklığı büyüktür demektir. Norman'a göre kusursuz bir arayüz yapmak mümkün olmadığı gibi hata yapmakta kaçınılmazdır. Bu yüzden tasarımda hata yapma durumunu minimuma indirmek ve hatayı düzeltmeye olanak tanımak gerekmektedir (Çağıltay, 2011).

Norman, değerlendirme ve gerçekleştirme körfezlerindeki açıklığı en aza indirmek için 4 temel strateji belirlemiştir. Bunlar;

- **Görünürlük:** Kullanıcılar sistemin o anki durum hakkında bir fikir sahibi olabilmelidir. Örneğin sistem beklemede ya da bir şey yüklüyorsa, kullanıcıya bunların farkına varmasını sağlayacak bazı özellikler sunmalıdır. Kum saati görseli buna güzel bir örnektir.
- **İyi bir kavramsal model:** Tasarımcılar, kullanıcının sistemde gerçekleştirdiği eylemlerle ve bunların sonuçlarının gösterimi arasında tutarlı bir kavramsal model oluşturmalıdır. Başka bir deyişle, kullanıcı yaptığı işlemin sonucunun ne olacağını önceden tahmin edebilmelidir.

Örneğin, bir web sitesinde tıklanabilir bağlantılar alt çizgi ile ya da renklendirilerek gösterilmezse kullanıcılar bunu anlayamayabilir.

- **İyi eşleştirmeler:** Gerçekleştirilen işlemler ve bunların sonuçları arasındaki ilişki iyi eşleştirilmelidir. Kırmızı renk genelde uyarı/ikaz, yeşil renk ise kabul/onay anlamına gelmektedir.
- **Geri bildirim:** Kullanıcılar, yaptıkları işlemlerin sonucu hakkında eksiksiz ve sürekli bir geri bildirim alabilmelidir. Örneğin, kullanıcı bir dosya indiriyorsa, dosyanın ne kadarının indiği ve ne kadar zaman kaldığı bilgisi kullanıcıya sürekli bildirilmelidir (Çağıltay, 2011).

2.1.5. İBE'nin bilişsel boyutu. Ekrana verilen bilgilerin kolay hatırlanması için nasıl organize edilmesi gerektiği ve kullanıcıların bir uygulamayı kullanmada zorlanırken benzer başka bir uygulamayı kullandığında neden bu sorunu yaşamadıkları gibi konuları açıklayabilmek için etkileşimin bilişsel boyutu hakkında bilgi sahibi olmamız gerekmektedir.

İnsanın bilişsel yapısı hakkında birçok çalışma mevcut olup, bu konuda beynin nasıl çalıştığını inceleyen farklı model ve yaklaşımlar bulunmaktadır. İBE için hangi modelin doğru olduğundan daha çok, bu modellerin etkileşim, arayüz tasarımı ve kullanılabilirlik testleri açısından nasıl kullanılabileceği konusu önemlidir. İBE alanında çalışan bir kişi için, genel kavramların ve terimlerin bilinmesi, beynin bilişsel yapısının ve kısıtlılıklarının genel olarak anlaşılması ve bunlar ile etkileşimin nasıl açıklanabileceğinin bilmesi yeterli olmaktadır. Bu kişiler için, algısal işlemci, kısa ve uzun süreli bellek, otomatik ve kontrollü işlemler, zihinsel gösterim, kodlama mekanizmaları, zihinsel model ve dikkat mekanizmaları/kaynakları gibi sistemlerin birbirleri ile ve dış dünya ile nasıl etkileştiklerinin bilinmesi gereken en önemli konular arasındadır (Çağıltay, 2011).

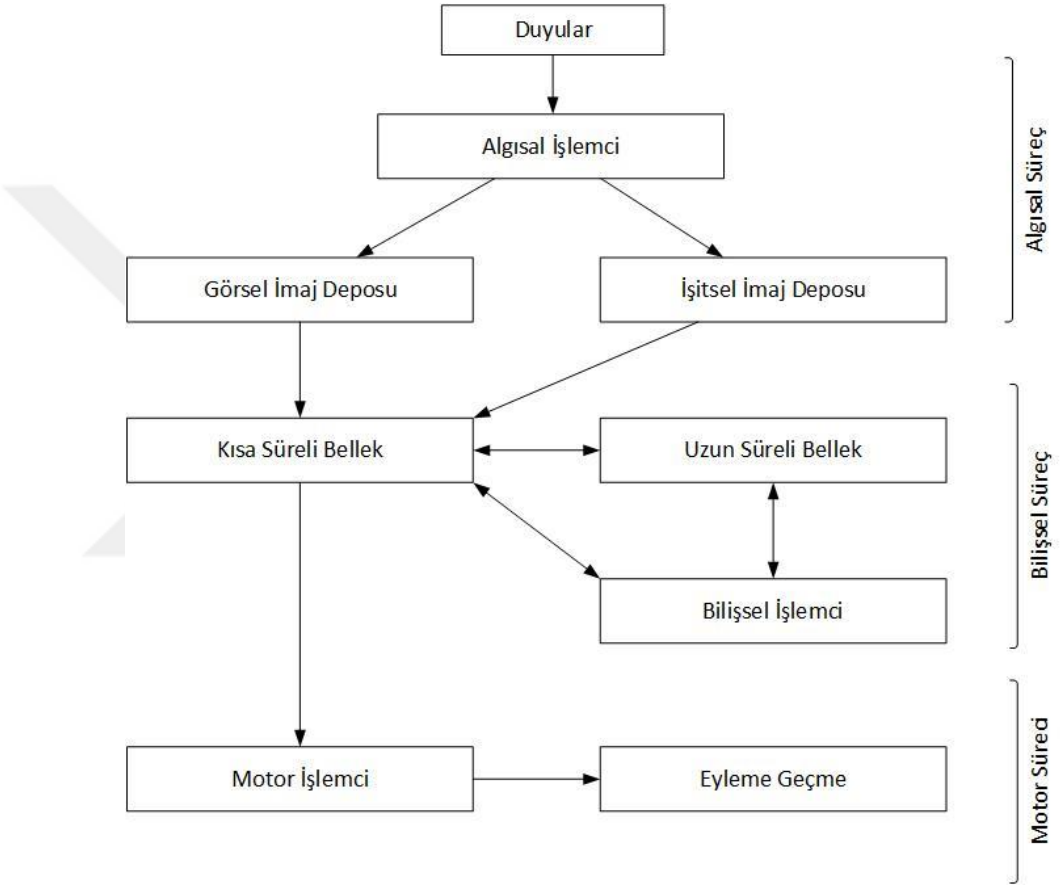
Bilgisayardan kullanıcının algı sistemlerine iki temel bilgi akışı olmaktadır. Birincisi ve en önemlisi, görsel bilgi akışı, ikinci olarak ise işitsel bilgi akışıdır. Kullanıcımızın arayüz ile olan etkileşimi sırasında gelen bilgiler önce göze ve kulağa ulaşmaktadır. Psikoloji alanında yaygın olarak kabul gören "Bilgi İşleme Modeli" yaklaşımına göre, gelen bilgiler beyinde derinlemesine işlenmeden önce, bir ön filtreden geçirilirler (Wickens, 1992). Algısal Ön İşlemci olarak adlandırılan bu basamak ekranında işlenmesine gerek duymadığı bilgileri görüş alanının içerisinde olmasına rağmen ya göz ardı eder ya da algılamaz. Örneğin, kişinin o an bakmakta olduğu hesap bakiye bilgisi, daha sonraki işlemleri için gerekli bir bilgi ise, kullanıcı onu kısa süreli belleğin içinde tutacaktır. Bilimsel çalışmalar ve teknolojik gelişmeler, görsel algının nasıl gerçekleştiği, bilgilerin beyinde nasıl kodlanıp geri çağrıldığı ve nasıl unutulduğu, insanların zihinsel sınırlılıklarının neler olduğu gibi konularda önemli bulgular sunmaktadır. Bu bulgulardan, bilişsel yük kuramına göre, beynimizdeki kısa süreli belleğin kapasitesi ve bilgilerin burada tutulma süresi sınırlıdır (Kalyuga, Chandler & J.Sweller, 2001). Kısa süreli bellek üstünde yapılan çalışmalar, burada tutulan bilgilerin ortalama 18 saniye yaşam ömrü olduğunu, bu bölgede bir anda ortalama 7 birim bilginin tutulabileceğini, zaman içinde çevreden gelen diğer bilgiler ve geçen süre nedeniyle bu bilgilerin silineceğini göstermektedir (Miller, 1956).

İBE alanında kısa süreli belleğin çalışmasının anlaşılması çok önemlidir. Çünkü kullanıcılar bilgisayar arayüzleri ile etkileşimleri sırasında ekranda bulunan bilgileri öncelikle kısa süreli belleğe yerleştirirler. Eğer arayüz çok karmaşıksa, anlamlı bir bilgi organizasyonu yapılmamışsa veya ekranlar/uygulamalar arasında tutarlılık bulunmuyorsa, kısa süreli belleğimiz verimli kullanılmayacaktır. Bunun sonucunda, kullanıcılar kullanım sırasında sistemde kaybolacak, yanlış seçimler yapacak ya da ne yapacaklarını bilemeyeceklerdir (Çağıltay, 2011).

Bunun önüne geçmenin yolu, kısa süreli bellekteki gerekli bilginin, uzun süreli belleğe aktarılmasıdır.

Şekil 2.3

İnsan İşlemci Modeli



Daha önce uzun süreli bellekte varolan bilgilerden yararlanan kullanıcı, hesabında olan paranın değerlendirmesini yapıp, bununla yatırım yapmaya ya da sistemden çıkmaya karar verir. Bu noktada, "Bilişsel İşlemci" alınan kararın sonucunu "Motor İşlemci" ye aktarır ve ondan gereğini yapmasını ister. "Motor İşlemci" fareyi tutan elin imleci sistemden çıkış yazısının üstüne götürmesini ve onu tıklamasını sağlar. Şekil 2.3'te gösterilen bu süreç İBE alan yazınında "İnsan İşlemci Modeli" (Model Human Processor) gösterimi ile açıklanır (Card, Moran ve Newell, 1983).

İBE açısından bakılırsa, eğer arayüz tasarımı bu süreci destekleyecek şekilde yapılırsa, kullanıcıların bilişsel sistemleri üstündeki yükü azaltmış ve onların yapacakları olası hatalar minimize edilmiş olacaktır. Aksi takdirde, kullanıcılar hata yapacaklar, kendilerine sunulan mesajları anlamayacaklar ya da yanlış yorumlayacaklardır. Bu da verimsiz bir ürün deneyimini kullanıcıya yaşatacaktır.

2.2. Kullanılabilirlik

Bu kısımda yazılımların da bir ürün olduğu düşünülürse kullanılabilirlik tanımlarından, kullanılabilirliğin tarihçesinden, kullanılabilirliğe etki eden faktörlerden ve çeşitli kullanılabilirlik değerlendirme yöntemlerinden bahsedilecektir.

2.2.1. Kullanılabilirlik tanımları.

"Kullanılabilirlik" kelimesini incelemek için Türk Dil Kurumu Güncel Türkçe Sözlük'e bakıldığında bir karşılığı bulunmamaktadır (TDK, 2015). Oxford sözlükte kullanılabilirlik araması yapıldığında ise, "kullanışlı (*usable*)" olarak yer verilmiştir. İngilizce karşılığı olan "usability" kelimesinin etimolojik açılımı usable + *-ity* olarak verilmiştir (Oxford, 2015).

ISO'a (1994) göre kullanılabilirlik; "Bir ürünün, belirli bir kullanım bağlamında, belirli kullanıcılar tarafından, belirli amaçları gerçekleştirmek üzere, etkin, verimli ve tatmin edici bir biçimde kullanılabilmesi'dir. ISO'nun yaptığı bu tanıma göre kullanılabilirlik; etkinlik, verimlilik ve memnuniyet olmak üzere üç temel esasa dayanmaktadır. *Etkinlik,* kullanıcıların amaçlarını ve görevlerini doğru ve eksiksiz olarak tamamlama seviyeleri, *Verimlilik,* amaçlara ve görevlere ulaşırken harcanan kaynakları, gayreti ve zamanı, *Memnuniyet* ise kullanıcıların sistem kullanımına ilişkin olumlu tutumları ve rahatlıkları ile ölçülmektedir.

Nielsen'a (1994a) göre kullanılabilirlik, ürünün kabul edilmesine etki etmekle birlikte; öğrenilebilirlik, hatırlanabilirlik, verimlilik, hata kontrolü ve memnuniyet olmak üzere beş ayrı özellikten oluşmaktadır. Benzer bir tanımı Thomas'ın (1998) yılında yaptığı çalışmasında da görebiliyoruz. Buna göre Thomas kullanılabilirlik özelliklerini görevler, süreç, çıktı olmak üzere üç ana kategoriye ayırmış; fonksiyonellik ve uyumluluğu görevler ile kullanım kolaylığıyla, arayüz, öğrenilebilirlik ve hatırlanabilirliği sistemin süreç ile etkinlik, verimlilik ve memnuniyet etmenlerini ise sistemin çıktılarıyla ilişkilendirmiştir. Goodwin (1987) ise kullanılabilirliği; bilişsel özellikleri öne çıkarak anlama, iletişim ve problem çözme etkinliklerine kullanıcıların gösterdikleri uyumluluk düzeyi olarak ifade etmiştir. Jeng (2006), kullanılabilirliği, en temel haliyle, sistem ve kullanıcının arayüz vasıtasıyla açık ve hızlı bir biçimde iletişim kurabilmesi şeklinde ifade etmiştir. Maguire'a göre (2001) kullanılabilir sistemler kullanıcıların tarafından daha kabul edilebilir görülüp verimliliği artırırken diğer taraftan hata yapma oranını ve öğrenme sürecine hız kattığı için kullanıcı desteğini de azaltmaktadır. Kullanıcılar iyi tasarlanmış ve kullanımı kolay sistemlere karşı pozitif bir tutum göstermekte ve daha fazla güven duymaktadır. Brinck, Gergle ve Wood (2001) kullanılabilirliği fonksiyonellik, kullanım verimliliği, öğrenim kolaylığı, hataları hoş görme ve kullanıcılar tarafından beğenilme gibi farklı unsurlarla açıklamıştır.

Kullanılabilirliğin aynı zamanda *kullanışlılık* ile de ilişkili bir kavram olduğuna dair çalışmalarda vardır. Bunlardan Nielsen'e (1994b) göre *kullanışlılık, kullanılabilirlik ve yararlılık* sistemin kullanıcılar tarafından kabul edilmesine etki eden ölçütlerdir. Landauer (1995) ise kullanışlılığı amaca uygunluk, kullanılabilirlik ise kullanım kolaylığı olarak ifade etmiştir. Davis'in (1989) geliştirdiği *Teknoloji Kabul Modeli'ne (TAM)* göre kullanım kolaylığı ve kullanışlılık, kullanıcıların yeni bir teknolojiyi kabul seviyeleri ve kullanımlarını etkileyen iki temel unsurdur.

2.2.2. Kullanılabilirliğin tarihçesi. Sanayi devrimi 1750-1850 yılları arasında yapısal ve düşünsel değişimlerin temelleri üzerinde gelişmiştir. Yapı itibariyle değişim geçirerek üretime artık makine, motor ve organizasyonu da katarak yeni bir sürece girilmiştir (Çetin, 2002). Peşi sıra gelen yeni icat ve buluşların gerçekleşmesiyle bu süreçte el aletleri ve insan yerini yavaş yavaş makinelere bırakmaya başlamıştır. Seri üretimin önü açılmış olup, tüm eski düzen üretim ve tüketim ilişkilerinde de büyük bir değişim meydana gelmiştir (Küçükkalay, 1997). Sanayi devrimi insan ile makine arasında bir fark bırakmamış, insanı, makineye tabi hale getirip onu kendine bir nevi köle yapmış denilebilir. Gelişen teknoloji ve geçen zamanla birlikte bu katı anlayış değişip insanı merkeze almış ve makineleri insan ihtiyaçlarına ve isteklerine göre karşılık verecek şekilde düşünsel bir şekilde değişim geçirmiştir. Örneğin kullanılabilirlik kavramının ortaya çıktığı zamanlarda basitlik ve kolaylık ile eş anlamlıydı. Daha sonra bu yaklaşımın değişmeye başladığını giderek gelişip günümüzdeki halini aldığını söyleyebiliriz (Schifferstein & Hekkert, 2008).

1970-80'ler de bilgisayar ve yazılımların üretim ve kullanımının yaygınlaşmasıyla hem donanım hem de yazılım boyutunda hatalar ve arayüz problemleri ile karşılaşılmaya başlanmıştır. Bu nedenle, kullanılabilirlik ile ilgili yayınlanan çoğu kaynak yazılım sistemleri kullanılabilirliği ile ilgidir. Foley ve Van Dam (1982), Rubinstein ve Hersh (1984), Eason (1984), Simpson (1985), Shneiderman (1987), Brown (1988), Dumas (1988) kullanılabilirlik ve yazılım sistemleri konusundaki ilk kaynak kitaplar olmuşlardır (Dumas, 2007).

1990'lardan sonra kullanılabilirlik prensipleri değişmeye başlamış, kullanıcı memnuniyeti, ürün çekiciliği ve kalite kavramları ortaya çıkmıştır. Özellikle kalite kavramı kullanıma uygunluk, kullanıcı tatmini, kullanımda kalite tanımları ile anılır olmuştur (Horie et al.,

2008). Nielsen, 1990 ile 1993 yılları arasında deneysel çalışmalar yapmış, bu çalışmalarda 249 adet kullanılabilirlik problemi ortaya çıkarmıştır.

Kullanılabilirlik kavramının yaygınlaşmasından sonra kullanıcı merkezli çalışmalara, kullanılabilirlik testi ve kullanıcı değerlendirmeleri de dâhil olmaya başlamıştır. İlk olarak Batı Avrupa' da ve Güney Amerika' da kullanılabilirlik testleri yapılmaya başlanmıştır. İleri teknoloji kullanan firmalar kendi kullanılabilirlik gruplarını kurmaya başlayarak çalışmalarına devam etmişlerdir (Dumas, 2007).

Zamanla gelişen kullanılabilirlik kavramı, artık ürün tasarımıyla birlikte düşünülür hale gelmiştir. Bu durum yazılım sistemlerine de yansımış ve ürün tasarımında vazgeçilmez bir noktaya gelmiştir.

2.2.3. Kullanılabilirlik faktörleri. Kullanılabilirlik tanımları ile beraber bir takım faktörlerden de bahsetmek gerekir. Bu faktörler kullanılabilirlik tanımlarındaki sözü edilen, kullanıcıların bir üründen beklentileridir diyebiliriz. Ancak bu faktörler ürüne ya da beklentilere göre değişebilir. Bu yüzden kullanıcıyı iyi tanımak ve onun beklentilerini bilebilmek çok önemlidir. Yukardaki verilen kullanılabilirlik tanımlarının çoğunda aşağıdaki ortak özellikler vurgulanmıştır;

- Etkililik.
- Memnuniyet.
- Verimlilik.

Bunların dışında kullanıcı, çevre ve ihtiyaçlara göre aşağıdaki diğer faktörleri de sıralayabiliriz:

Güvenilirlik; Ürünlerin kapasiteleri ve performansları belirtilen koşullar altında kullanılması durumunda kullanıcının ürüne güvenilirliği artacaktır. Ürünün kullanıcıya memnuniyet

sağlaması, yaşanılan problem ve arızaların çözümlenmesi, belli standartlarının olması ürünün güvenirliğini artıran unsurlardır (Bevan, 2000).

Uygunluk; Ürünün hedef kullanıcıya hitap edip edemeyeceği anlamına gelir. Kullanıcı kendi amaçları doğrultusunda üründen yararlanmak ister. Tasarımcı da yaptığı ürününün kullanıcılara uygun olmasını arzu eder ve kullanabilmesini tercih eder. Bu tasarımcı ve kullanıcı arasındaki karşılıklı bir beklentidir (Bevan, 2000).

Dayanıklılık; Ürünlerin genellikle bir kullanım ömürleri ya da kullanım süreleri bulunur. Bu süre içinde ürünün kullanıldığı zamandaki performansı ya da ürünün karşılaştığı fiziksel aşınmalara karşı fonksiyonlarını yitirmemesi gibi durumlar bu ürünü dayanıklı kılacaktır. Bu durum kullanıcı tercihlerine göre de değişebilir. Bazı kullanıcılar bir ürünü alıp kısa süre kullanıp değiştirmeyi tercih ederken başka bir kullanıcı bir ürünü uzun süre kullanmak isteyebilir. Üretim kapasitesi, maliyet gibi faktörler bunda etkili olmaktadır (Bevan, 2000).

Çekicilik; Ürünün sahip olduğu renkler ve tasarım detayları, onu diğer ürünlerden ayırıp kullanıcı tarafından daha çabuk dikkat çekmesini sağlayabilir. Ürün çekiciliği çoğu zaman kullanıcıyı etkileyen bir unsur olmuştur (Bevan, 2000).

Hata tahmini; Ürünü yaparken kullanıcıların ürün kullanımı sırasında meydana gelebilecek hatalarını tahmin edebilmek ve tasarımı bu yönde değiştirmek gerekir. Kullanıcıların uyum sorunu yaşayacağı problemlerin analizi yapılıp bunların düzeltilmesi yoluna gidilmelidir (Pheasant, 2003).

Uyum; Ürünün farklı ortamlarda kullanıldığı durumlarda oluşabilecek değişikliklere uyum sağlaması beklenir. Bu uyumlu kullanım sayesinde kullanıcı ile ürün arasında olumlu bir iletişim kurulur (Redish, 2007).

Yaratıcılık ve inovatiflik; Yenilikçi ürünleri seven bir kitleye yaratıcı ve inovatif ürünler tercih edilebilir gelebilir. Bu da kullanıcının o ürünle daha kolay ve olumlu iletişime geçmesine neden olabilir (Redish, 2007).

Fonksiyonellik; Ürünün kullanıcıya sunduğu özelliklerin ve işlevlerin kullanılabilmesidir.

Sürdürülebilirlik; Ürünün kapasitesi zamanla değişebilir. Bu değişimler, düzeltme, geliştirme ya da adaptasyon olabilir (Redish, 2007).

Hata sıklığı; Ürünlerin kullanılırken olabildiğince az hata sıklıkları olması kullanıcıyı olumlu etkileyecektir. Hata oluş sıklığı ile memnuniyetsizlik doğru orantılıdır. Bu durum ürünü kullanışsız kılabilir (Redish, 2007).

Kullanıcı güvenliği; Kullanıcı ürünü kullanımına devam ederken kendini güvende hissetmek ister. Böyle bir durum, kullanıcının ürün ile bağ kurmasına sebep olur (Kesseler & Knapen, 2006).

Öğrenebilirlik; Ürünün sahip olduğu fonksiyonların çabuk öğrenilmesi, onun verimli kullanılmasına yardımcı olduğu gibi aynı zamanda kullanıcının o ürünle vakit kaybetmemesini sağlar.

Yönetilebilirlik; Ürünün tutarlı olması, açıklayıcı mesajlar verip kullanıcı ile iletişime geçmesi, kullanıcının ürünü kullanırken yetersiz kalmaması, kullanıcının ürünü kendine göre kullanmaya başlayabilmesi, bütünleşmesi gibi durumların hepsi kullanıcı tarafından ürünün yönetilebilirliği anlamına gelmektedir (Bevan, 2000).

Anlaşılabilirlik; Ürün ilk kez deneyimleyecek olan kullanıcılarında ürünü kolayca anlayabilmesi ve uyum gösterebilmesi gerekir. Ayrıca ürünü her kullanıcının oldukça anlaşılır, kolay kullanılabilir olması beklenir (Bevan, 2000).

Sağlayıcılık; Üründen beklenen hedef ve amacın gerekli kontroller yardımı ile gerçekleştirebilmesidir (Moore, 1997).

Yönlendirme; Üründe bulunan bir kontrolün sizi bir sonraki hareketinize doğru yönlendirmesi, yani bir sonraki hareketi doğru algılayıp o yönde hareket etmenizi sağlaması gerekir. Buna günlük olarak sıkça kullanılan ocaklar örnek verilebilir. Kullanıcılar ocağı açtıklarında bir sonraki adımda ocağı kapatacakları zaman düğmeyi tersi yöne çevireceklerini kolayca tahmin edebilirler. Dolayısıyla ocakların kullanıcıları doğru yönlendirdiklerini söyleyebiliriz (Moore, 1997).

Kontrol duygusu; Kullanıcı ürün kullanımı sırasında bir yönlendirme bekler ve bu yönlendirme doğrultusunda hareket eder. Bu yönlendirmeyi eylemini kolay ve anlaşılır bir şekilde kullanıcıya aktarabilmesi gerekir (Moore, 1997).

Tutarlılık; Ürünün arayüzündeki kontrollerin benzerliklerinin anlaşılır olması gerekir. Örneğin bir müzik sisteminde radyo, cd ve diğer ayar düğmelerinin doğru anlaşılır yerlerde olması gibi. Bu durum genelde çok fazla tercih edilen ürünlerde beklenen bir istisna, ayrıcalıktır (Moore, 1997).

Minimalizm; Kapsam olarak daha az özelliğin olması amacın daha net ve hızlı anlaşılmasını sağlar. Bu şekilde kolay ve başarılı bir kullanım gerçekleştirilebilir (Moore, 1997).

İteratif tasarım; Tasarımcılar ürünlerini gerçek kullanıcılarla test ettiklerinde kullanıcı tereddütleri ve hataları ortaya çıkacaktır. İteratif tasarımda bunu ortadan kaldırmak ve tekrar etmemesini sağlamak mümkündür (Moore, 1997).

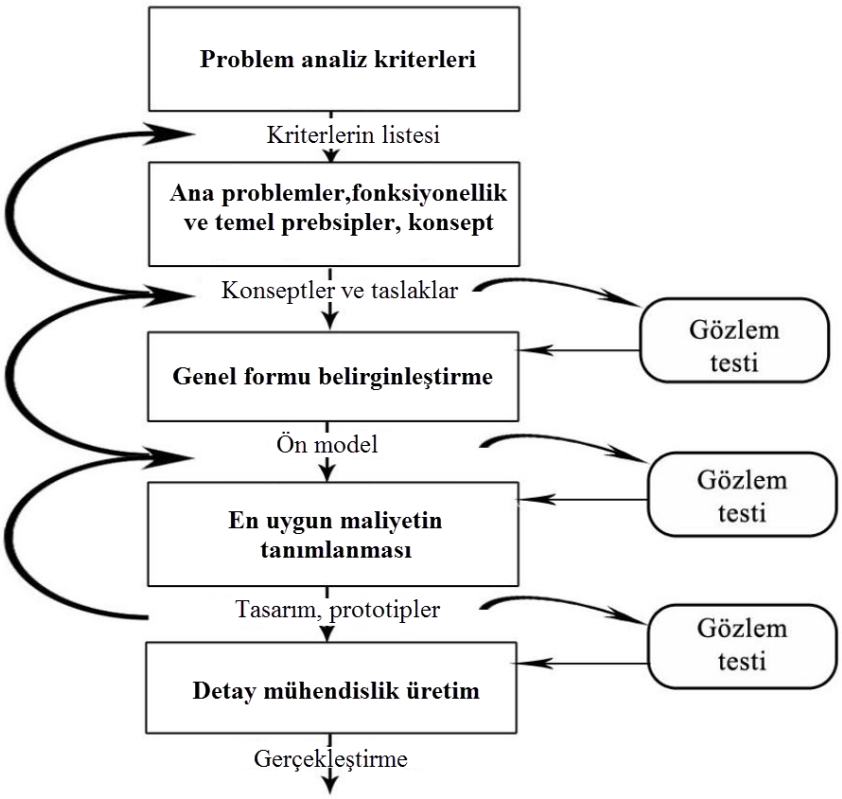
Sonuç olarak bu faktörlerden sadece bir kavrama dayandırılmadan hepsinin birlik ve uyum içinde kullanılmasına çaba gösterilmelidir (Redish, 2007). Bu sayede o ürünün daha çok tercih edilmesi sağlanabilir.

2.2.4. Kullanılabilirliğin değerlendirilmesi. Arayüzlerin kullanılabilirlik değerlendirmeleri için birçok sayıda yöntem ve teknik var olup bunlar temel olarak; Kullanılabilirlik Testi, İnceleme Yöntemleri ve Sorgulama Yöntemleri şeklinde üçe ayrılmaktadır.

2.2.4.1 Kullanılabilirlik testleri. Bir bilişim sistemi kullanıcısının o sistemi ne derece etkin, verimli ve memnun kalarak kullanıp kullanmadığını ortaya çıkarmak için kullanılan araştırma yöntemlerine ise "Kullanılabilirlik Testleri" denilmektedir (Çağıltay, 2011). En popüler yöntemlerden biri olan kullanılabilirlik testleri; arayüzlerin gerçek kullanıcıları temsil eden kullanıcılar ve gerçek görevler yoluyla test edilmesine dayalı sistematik ölçüm yöntemlerini içermektedir (Dumas & Redish, 1993; Rubin, 1994). Test yürüten kişi, arayüzün etkinliğini ölçmek için kullanıcıların sistemle olan etkileşimlerini ve tutumlarını kontrollü olarak gözleyip veri toplamaktadır (Corry, Frick & Hansen, 1997; Dumas & Redish, 1993). Bu testler karmaşık testlere göre daha az kullanıcı ile yapılan nitelikli ve kalıcı çalışmalardır. Bazı firmalar kullanılabilirlik testini, ürünlerin kârını artırmak ve kolaylaştırmak için yapmaktadırlar (Rubin & Chisnell, 2008).

Şekil 2.4

Kullanılabilirlik Testi Süreci (Buurman, 1997)



Kullanıcıları gözlemleyerek elde edilen fikirler, kazaların, hatalı kullanımın ya da yaralanmaların araştırılması, kullanımda başarı veya başarısızlıkların nedenlerinin araştırılması, performans testleri, kullanıcının ürün ile ilgili algısal geribildirimleri gibi önemli sonuçlara odaklanılarak olası problemlerin erken çözülmesini sağlamaya yardımcı olur ve kullanıcı memnuniyetinin artmasını sağlar (McCormic & Sanders, 1993).

Kullanılabilirlik testlerinin yapılış amaçlarını Rubin ve Chisnell (2008) şu şekilde sıralamaktadırlar;

- Tasarımın amacı hakkında bilgi toplama analiz etme ve bunları düzenlemek;
- Olabilecek tasarım problemlerinin engellenmesi; kullanıcılar ile doğru iletişim kurulmasını sağlayabilmek ve ürün hedeflerine ulaşabilmelerini sağlayabilmek;

- Birçok açıdan elde edilen kârı artırmak; firma ya da tasarımcıların zaman ve maddi açıdan kazanç elde etmelerini sağlayabilmek.

Head'e (1999) göre kullanılabilirlik testlerinde en fazla 3-5 kullanıcı, her bir görev için 4-5 dakika ve her bir test için en fazla 1 saat süre yeterlidir. Literatürde kullanılabilirlik testi için yeterli katılımcı sayısı ile ilgili farklı görüşler yer almaktadır. Bazı araştırmalara göre Chisman, Diller ve Walbridge (1999), 8 kullanıcı test için yeterli olurken, kimilerine göre Dickstein ve Mills (2000) ise 8 ile 12 katılımcı yeterli bulunmaktadır. Nielsen (2015b), katılımcı sayısı ile test sırasında tespit edilen problemler arasındaki ilişkiyi araştırmıştır. Araştırma sonucuna göre 15 kullanıcı ile %100, sekiz kullanıcı ile %90, 5 kullanıcı ile %80 oranında problemlerin tespit edilmiştir. Bu sonuçlara göre Nielsen orta büyüklükte bir proje geliştirirken 3-5 kişilik homojen bir kullanıcı grubu ile test yapılması yeterli olduğunu ifade etmiştir.

Dumas ve Redish (1993), kullanılabilirlik testlerinin karakteristiklerini şu şekilde sıralamıştır:

1. Temel amaç ürünün kullanılabilirliğini geliştirmektir;
2. Test katılımcıları gerçek kullanıcıları temsil eden kişilerden oluşmaktadır;
3. Test sırasında arayüzle ilgili gerçek görevler kullanılmaktadır;
4. Test sırasında katılımcıların davranışları gözlenmekte ve yorumları kaydedilmektedir;
5. Test aracılığı ile gerçek problemler tespit edilebilmektedir.

Kullanılabilirlik testlerinde en çok kullanılan tekniklerin başında *Sesli Düşünme* tekniği gelmektedir. Sözlü Protokol Analizi olarak da bilinen bu tekniği, başta bilişsel psikoloji olmak üzere, bilgi sistemleri, yapay zekâ ve İnsan-Bilgisayar Etkileşimi gibi alanlarda sıklıkla kullanılan bir tekniktir. Ericson ve Simon (1993), tarafından geliştirilen sözle ifade etme modelinde, araştırma esnasında algısal süreçlerle ilgili gözlem verilerine derinlik katmaktadır. Sistemle yapılan gerçek bir etkileşimin gözlendiği bu teknikte, kullanıcılardan tanımlanmış bir görevi

yerine getirirken, düşüncelerini eşzamanlı olarak sesli olarak ifade etmeleri istenmektedir (Boren & Ramey, 2000). Bu şekilde, kullanıcıların sistemle ilgili algısal modeli nasıl anladıkları ve şekillendirdikleri incelenip, arayüz ile ilgili varsayım, çıkarım, yanlış anlama ve problemleri ortaya çıkarmaları sağlanmakta, insan algısı ve bilişsel süreçler ile ilgili önemli veriler elde edilmektedir (Ericson & Simon, 1993).

Sesli düşünme tekniğinin avantajları,

- Bu süre içinde kullanıcılar düşüncelerini söyledikleri için, testin sonunda tekrar sormaya gerek kalmayabilir;
- Bu teknik süresince kullanıcılar test yoğunlaştıkları için, çalışmanın ve konuşmanın akışına kendilerini kaptırabilirler;
- Bu teknik yeni bir fikrin oluşmasına da neden olabilir.

Dezavantajları ise,

- Sistem bazı kullanıcılar tarafından dikkat dağıtıcı ve doğal bulmayabilir;
- Bu teknik kimi zaman da süreci yavaşlatabilir, dikkati dağıtabilir. Böyle bir durumda onlardan test süresince daha dikkatli olmaları istenebilir fakat bu da sürece farklı yansıyabilir (Rubin & Chisnell, 2008).

Kullanılabilirlik testlerinin birçoğu belirli bir disiplin ve yüksek teknolojili ekipmanların olduğu laboratuvar ortamlarında yapılabilir. Ancak bundan daha önemli olan bu test sürecinin kendisidir. Bunun için pahalı bir laboratuvar gerekmeyebilir. Amaç; hedefi yerine getirebilme ve kullanıcılar ile iletişim kurabilmektir. Bunun içinde test organizasyonu iyi planlanmalı,

kullanılabilirlik ve tasarım faaliyetleri yöntem ve teknikleri hakkında temel bilgiye sahip olmak gerekir (Rubin & Chisnell, 2008).

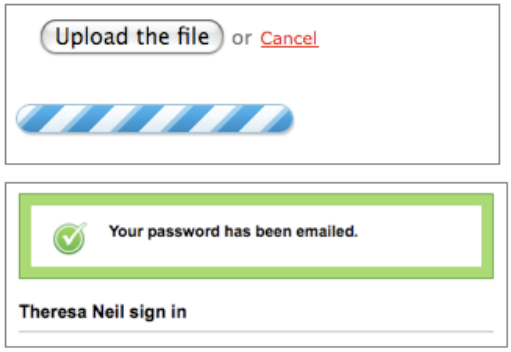
Diğer taraftan, Rubin ve Chisnell (2008), bu konudaki deneyimlerine dayanarak her kullanılabilirlik testinin laboratuvarda yapılmaması gerektiğini söylüyor, çünkü bazı testlerin laboratuvar ortamında pratik ve kullanışlı olamayabileceğinden bahsediyor. Bu fikri destekler nitelikte Barnum ve Dragga (2001), ürünlerin gerçek ortamlarında kullanılmasını belirtip aksi takdirde başka test ortamı seçeneklerini şu şekilde sıralıyor;

- **Laboratuvarda**; kendi laboratuvarınızda ya da kiralanacak bir laboratuvarda;
- **Konferans salonu**; test için bir oda kiralanarak yaparak;
- **Ürünün kendi ortamında**; kullanıcıların ürünü kullandıkları gerçek ortama giderek;
- **Uzaktan test ederek**; kullanıcıları uzaktan gözlemleyerek.

2.2.4.2 İnceleme yöntemleri. Arayüz tasarımı ve testinde takip edilen ve çok yaygın diğer bir yöntem ise arayüzün o alanın uzmanlarınca değerlendirilmesidir. Uzman değerlendirmeleri içinde en yaygını, iyi bir arayüz tasarımında olması gereken özellikleri veren sezgisellerin (heuristics) kullanılmasıdır. Sezgiseller genelde ortamdan (donanım veya yazılım) bağımsız olup kullanılabilirliği geliştirmeye yöneliktir. Kullanıcı arayüz tasarımı için kullanılan en popüler sezgisel rehberlerinden birisi, Jacob Nielsen tarafından önerilmektedir. Bu çalışmada da Nielsen'in sezgiselleri kullanılmıştır. Nielsen'in, 10 Kullanılabilirlik Sezgiseli (Nielsen's Ten Usability Heuristics) adı verilen bu rehbere (2015) göre, kullanıcı arayüzleri tasarımında aşağıdaki noktalara dikkat edilmeli ve kullanılabilirlik bu çerçevede sorgulanmalıdır.

2.2.4.2.1 Nielsen'in Sezgiselleri

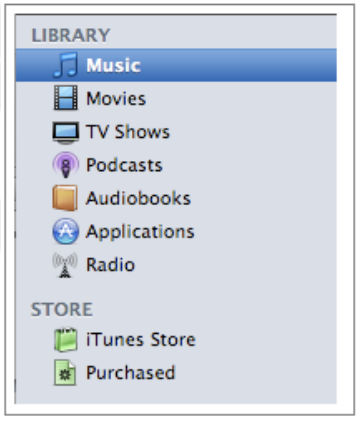
1.Sistem Durumunun Görünürlüğü



Sistem, kullanıcıyı sürekli olarak o anki durumunun nasıl gittiği hususunda sürekli olarak uygun dönütlerle bilgilendirmelidir.

Sağdaki ekranda dosya gönderme süreci ve e-postanın gönderilmesi hakkında kullanıcıya anlık bilgi veriliyor.

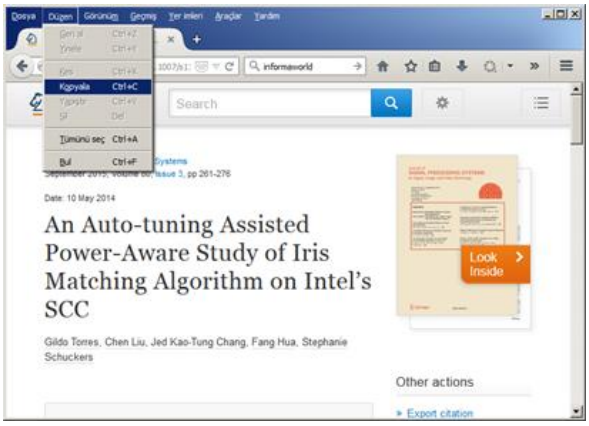
2. Sistem ile Gerçek Dünyanın Eşleşmesi



Sistem, kullanıcıların anlayabileceği dilde kullanmalı, terimler, kelimeler ve kavramlar kullanıcıya yabancı gelmemelidir. Bilgilendirmeler kullanıcıya mantıklı bir şekilde verilmelidir.

Apple'in iTunes yazılımındaki ekran görüntüsünde müzik, film, dizi gibi günlük hayat kavramı yine bunları çağrıştıracak simge veya ikonlarla gösterilmiştir.

3. Kullanıcı Kontrolü ve Özgürlüğü



Kullanıcılar sık sık sistem fonksiyonlarının seçiminde hata yaptıklarında çıkmak için açıkça belirtilmiş bir "acil çıkış"a ihtiyaç duyarlar. Geri alma (*undo*) ve tekrar yapma (*redo*) seçenekleri bu nedenle kullanıcıya sunulmaktadır.

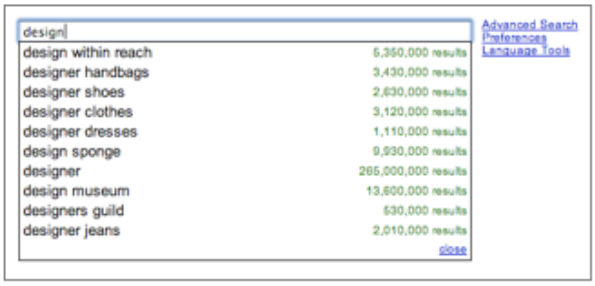
Sağdaki tarayıcı ekranında yapılan işlemlerle ilgili geri alma ve yineleme işlevleri kullanılabiliyor.

4. Tutarlılık ve Standartlar

Kullanıcılar farklı kelimelerin, durumların ve eylemlerin aynı anlama gelip gelmediğini düşünmek konusunda kararsız kalmamalıdır. Sistem kendi içinde bütünüyle tutarlı olmalıdır.

Microsoft Office ürünlerinin çoğunda menü tasarımı ve menü seçenekleri aynıdır, tutarlıdır.

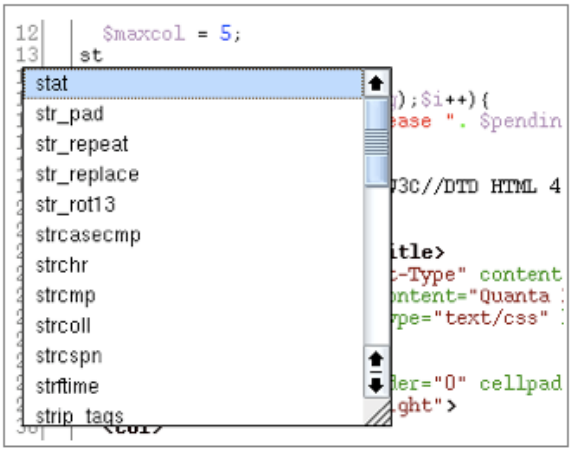
5. Hataları Önleme



Dikkatli bir tasarım ile kullanıcıların hata yapmaları engellenebilir.

Google arama motorunda aranan kelimeler hakkında otomatik tavsiyeler çıkması hatalı girişi önlemiş oluyor.

6. Hatırlama Yerine Tanıma

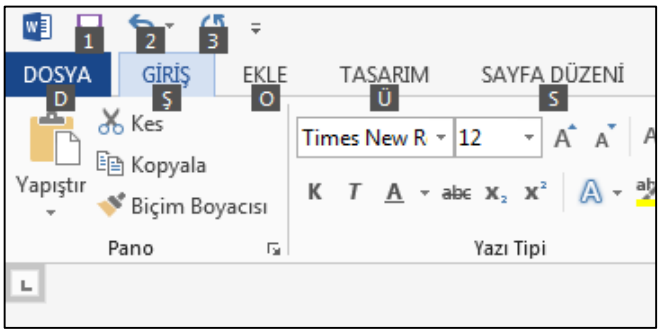


Nesneler, aktiviteler, seçenekler, gerekli talimatlar görünür ve ulaşılabilir yerlerde olmalıdır. Kullanıcı diyaloğun bir bölümünden diğerine geçişlerde, bir önceki bölümü hatırlamak zorunda bırakılmamalıdır.

Sağdaki ekranda girilecek kodları hatırlamak yerine o kodun birkaç harfini girmek yeterli

oluyor ve ilgili kodu hatırlatan bir liste çıkıyor.

7. Esneklik ve Kullanım Verimliliği



Yeni kullanıcılar tarafından görülemeyen hızlandırıcılar (kısayollar) kullanılmalıdır. Deneyimli ve deneyimsiz kullanıcılar farklı kullanım davranışları sergilerler. Her iki grubunda yararlanması için, uzman kullanıcılara yönelik, etkileşimi daha kısa yoldan ve hızlıca sağlayan teknikler kullanılmalıdır. Kullanıcılara çok sık kullandıkları fonksiyonları kendilerine göre özelleştirebilmeleri için seçenekler sunulmalıdır.

Sağdaki ekranda acemi kullanıcılar fareyi hareket ettirerek sekmelerde dolaşırken, deneyim kullanıcılar klavye kısayolları kullanarak istediği işlemi yapabiliyor.

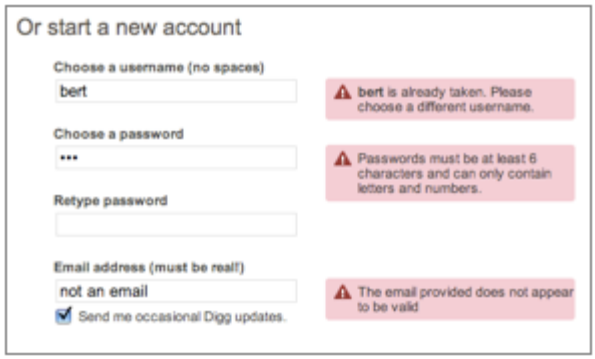
8. Estetik ve Sade Tasarım

Kullanıcı ile kurulan diyaloglar ilgisiz ya da pek gereksiz bilgiler içermemelidir. Bir diyaloğa giren her ek bilgi, daha önemli bilgilerin görülmemesini sağlayıp, karışıklık oluşturur. Eğer bir bilgi ya da resim arayüzden kaldırıldığında kullanıcı etkileşiminde bir şey değişmiyorsa, o unsur gereksiz demektir. “Gereksiz ise kullanma” kuralı uygulanmalıdır.

Web dünyasında belki de en sade tasarıma sahip sitelerinden birisi Google’dır. Kullanıcının dikkati sadece veri girişine odaklanmıştır.

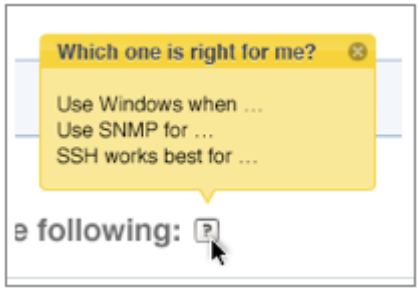
9. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma

Hataların geri bildirimleri, anlaşılır bir dilde (teknik olmadan) olmalı, sorunu açıklamalı ve yapıcı çözüm önerisi sunabilmelidir. Yandaki ekranda ilgili kutucuklara girilen veriler için kutucukların sağındaki alanlarda bazı kurallar verilmiş. Bu şekilde hatalı girişi tanıma ve önlemeye yardımcı olunuyor.



10. Yardım ve Belgeleme

Kullanıcıya yardım etmesi bakımından bir yardım sistemi sunmak gerekli olabilir. Yardım sisteminde aranan bilgiyi bulmak kolay olmalı, yardım dokümanı kullanıcının problemine odaklanmalı, çözümleri sunarken adım adım bunları göstermelidir. Yandaki ekranda "?" ikonu üzerine fareyi getirince o maddeyle ilgili bilgi ayrı bir pencerede belirmekte ve kullanıcı istediği yardım ya da dokümana hemen ulaşabilmektedir.



Uzmanlar, arayüz değerlendirmelerinde kullanılabilirlik testleri için, sezgiselleri oldukça yaygın olarak kullanmaktadırlar. Sezgiseller çok hızlı değerlendirme yapılmasına izin verirken ayrıca özel bir donanım da gerektirmemektedir (Çağıltay, 2011).

Uzman temelli kullanılabilirlik testi isminden de anlaşıldığı üzere, kullanılabilirlik uzmanları tarafından yapılır. Sezgisel tabanlı değerlendirmeyi, kullanılabilirlik alanında uzman birden fazla kişi tarafından yapmalıdır. Uzmanlar, bu sayede, yaptıkları değerlendirmeleri karşılaştırıp, yorumları arasında bir tutarsızlık olup olmadığını tespit ederler. Uzmanlar mümkünse yapacakları uygulama konusunda önceden bilgi sahibi olmalıdırlar (Çağıltay, 2011).

Bir diğer incelemeye dayalı yöntem olan *Bilişsel Gezinti* yönteminde bilişsel araştırmalar ışığında, kullanıcıların arayüzle yaptıkları etkileşimler canlandırılmaya çalışılmaktadır (Horn, 2015). Canlandırmada değerlendirme uzmanları, kullanıcıların yerine geçerek arayüz tasarımının her bir aşamasını tek tek dolaşıp ve kullanıcılar için problem oluşturacak noktaları bulmaya çalışmaktadırlar (Wharton, Rieman, Lewis & Polson, 1994). Bu yöntem, tıpkı sezgisel değerlendirmeler gibi, sürecin basitliği ve herhangi bir araç ve gereç ihtiyaç duyulmaması sebebiyle çok kolay gerçekleştirilen bir yaklaşımdır. Bilişsel Gezinti yaklaşımı, ürünün geliştirilme safhasında (süreç içi) yapılıp tasarımcılara geri bildirim sağlamaktadır (Çağıltay, 2011).

Yukarıda belirtilen inceleme yöntemlerden başka bazı yöntemler daha vardır, bunlar; Formel Kullanılabilirlik İncelemeleri, Özellik İnceleme, Kılavuz Denetim Listeleri şeklindedir.

Nielsen (1994b), çalışmasında incelemeye dayalı yöntemleri düşük maliyetli ve hızlı değerlendirilmesinden dolayı indirimli kullanılabilirlik yöntemleri (Discount usability methods) olarak da ifade etmektedir. Ancak hızlı değerlendirilmesine karşın, sorgulama yöntemleri ve kullanılabilirlik testleri ile kıyaslandığında, bu yöntemlerin çeşitli sınırlılıkları vardır. Değerlendirmeler gerçek kullanıcılar yerine uzmanlar tarafından yapıldığı için problem tespitlerinde de yanılgılar olabilmektedir.

Çağıltay'a (2011) göre arayüzlerin geliştirilmesi ve değerlendirilmesinde inceleme yöntemlerinin faydalı olmasına karşın, genel olmalarından dolayı yorum açıktır. Örneğin, arayüzde tutarlığın/uygunluğun nasıl sağlanacağı, geri bildirimin (dönüt) ne şekilde verileceği, bir hata mesajında hangi özelliklerin olması gerektiği gibi birçok konuda, sezgisel rehberlerde de yeterli ve belirleyici bilgiler bulunmaz.

2.2.4.3 Sorgulama yöntemleri. Sorgulamaya dayalı yöntemleri de, sistemin gerçek kullanıcılarından veri elde etmek için uzmanlar tarafından gerçekleştirilen çalışmalardır. Bunlardan bazıları*; Alan Gözlemi, Görüşme ve Odak Grupları, Anketler* ve *Bağlamsal Sorgulama* şeklindedir.

*Alan Gözlemi'*nde, gerçek kullanıcılar, uzmanlar tarafından gerçek işlem süreçleri esnasında yerinde gözlenmekte ve kullanıcıların sistemi nasıl kullandığına ilişkin bilgiler elde etmeye çalışmaktadırlar. Kullanıcıların kendi normal çevrelerinde (iş, ev, bar vs.) sistemi kullanarak amaçlarına ne kadar yaklaşıldığına bakılır (Rubin & Chisnell, 2008).

*Görüşme ve Odak Grupları'*nda ise gerçek kullanıcıların arayüze ilişkin varolan problemleri, önerileri, tercihleri gibi çeşitli konularda doğrudan bilgi alınmaktadır. Bu teknikte kullanıcılara ürünler hakkındaki fikirleri sorulur. Ancak burada kullanıcıların ürün kullanımı ile ürün hakkındaki fikirleri arasında farklar çıkabileceği için, bu teknik ile kullanılabilirlik tam olarak anlaşılamayabilir (Rubin & Chisnell, 2008).

Bir diğer sorgulama yöntemi olan *Anketler,* kullanılabilirlik değerlendirmelerinde çokça kullanılan önemli araçlardır. Bu teknikle kullanıcıların, ürün kullanma sürecinde gösterilen performansları ve deneyimleri hakkında bilgi alınmasını sağlar. Anketler de önemli olan husus, kullanıcıların mümkün olduğu kadar doğru yanıtlar verebileceği, ilgisinin kopmayacağı şekilde testi tasarlayabilmektir. Anketlerde belli bir kesimi ayrıntılı bir şekilde araştırmak yerine daha geniş bir kesimi ele almak gerektiğinden oldukça kalabalık kullanıcı kesimi ile yapılabilir. Bu teknik kullanıcıların ürün memnuniyetini öğrenmek için kullanılabilir. Anket tekniği kullanılabilirlik testlerindeki gibi izleme olanağı sağlamadığından kullanılabilirlik testlerinin yerini alamaz (Rubin & Chisnell, 2008). Kullanılabilirlik anketleri, sistem tasarımcıları

tarafından hazırlanabilirken, yazılım ve arayüz değerlendirmeleri için geliştirilen standart anketler de vardır. Bunlardan bazıları şunlardır:

- John Brooke (1996) tarafından geliştirilen System Usability Scale– SUS.
- Software Usability Measurement Inventory – SUMI.
- Maryland University'in insan-bilgisayar etkileşimi laboratuvarında (1988) geliştirilen Questionnaire for User Interface Satisfaction – QUIS.
- End-User ComputerSatisfactionInstrument – EUCS.
- Jim Lewis (1991) tarafından geliştirilen After-Scenario Questionnaire – ASQ.
- Post-StudySystem Usability Questionnaire – PSSUQ.
- Web Site Analysis and Measurement Inventory – WAMMI.
- Jim Lewis (1995) tarafından geliştirilen Computer System Usability Questionnaire –CSUQ.

2.3. Tasarım

Kullanılabilir bir ürün tasarımı, kullanıcı araştırması, tasarım araştırması ve devamında tasarım sürecinde kullanıcı deneyimlerinin değerlendirilmesi ile mümkün olabilir (Telek, 2013). Bu kısımda farklı tasarım ve arayüz tanımlarından, arayüz tasarımının amaçlarından, öğretim yazılımlarında dikkat edilmesi gereken tasarım ilkelerinden ve ekran özelliklerinden bahsedilecektir.

2.3.1 Tasarım ve arayüz. Tasarım temel olarak bilgilerin örgütlenmesidir denilebilir. Bu bilgiler grafik biçiminde, metinsel veya sayısal şekilde olabilir. Tasarımcının işi ise, kullanıcılar tarafından anlaşılabilecek örgütlenmeyi gerçekleştirmektir (Smith, 2015). Tasarım algı ile kavram arasında köprü görevi görür. Genel anlamda tasarım bilgi edinme öğesidir (Medyanadolu, 2015).

Buradan da anlaşılacağı üzere, verilmek istenen kavram bilgisini grafik, ses, metin gibi unsurları mümkün olduğu kadar en iyi forma sokup insan algısına yakınlaştırma işlemine tasarım diyebiliriz. Kazu ve Yavuzalp'in yaptığı bir araştırmada (2008) öğretmenler, öğretim yazılımlarının materyal tasarım ilkelerine uygun, rahat ve kullanımı kolay olması gerektiğini ifade etmişlerdir.

Kullanımı kolay yazılımlar çok kısa sürede oluşturulmamaktadır. Yazılım tasarımının ilk aşamasından son aşamasına kadar yazılımın potansiyel kullanıcılarına odaklanması gerekmektedir. Her aşamada kullanıcıların hangi tasarımlardan hoşlandığı ve kendilerini rahat hissettikleri kontrol edilmelidir. Yazılım geliştirme işi bir ekip işidir. Bunun için çok farklı alanlardan kişilerin oluşturduğu proje takımı kurulmalıdır. Kullanıcı merkezli arayüz takımının ilk aşamada ürünü kullanacakların özelliklerini belirlemesi gerekmektedir (IBM, 2015).

Tasarımın özel bir biçimi olan arayüz tasarımı diğer tasarım türlerinde olduğu gibi tasarımcı bilgiyi alır ve anlamlı bir biçimde örgütler. Bu bilgiler genelde ihtiyaç tanımlaması (analizleri) adı verilen dokümanlardan sağlanır. Tasarımcı ürünü kullanacak kitlenin ihtiyaçlarını en iyi şekilde karşılayabilmesi için bu bilgilerden yararlanır (Smith, 2015).

2.3.2 Arayüz tasarımının amaçları. Smith'e (2015) göre ara yüz tasarımının temel hedefleri, öğrenme zamanı, performans hızı, kullanıcı kaynaklı hataların oranı, yazılım kullanımını hatırlama ve memnuniyettir.

1. Öğrenme Zamanı

Yeni bir kullanıcının yazılımı öğrenmede geçirdiği süreye öğrenme zamanı denilmektedir. Öğrenme zamanı genelde yazılımın işlevselliğine (fonksiyonellik) bağlıdır. Çok karmaşık ve kötü bir şekilde tasarlanmış olan yazılımın öğrenilmesi zor olacak ve daha uzun

sürecektir. Diğer taraftan aynı işlemi gerçekleştirmek için farklı terim kullanımı da öğrenilme süresini etkileyecektir. Çünkü kullanıcılar bu terimleri öğrenmek zorunda kalacaktır.

Yazılımlarda öğrenme süresi önemli olmasının iki nedeni vardır: Birincisi, öğrenciler yazılımları sınırlı bir süre için kullanırlar. Bu nedenle de yazılım kullanımının mümkün olduğu kadar kısa sürede öğrenilmesi gerekmektedir. İkincisi, öğrenciler materyalleri yeni bilgiler öğrenmek için kullanırlar. Yazılımı kullanmayı öğrenme, bilgiyi öğrenme sürecine eklenmektedir. Öğrenme esnasında yazılımla ilgili çıkabilecek zorluklar çözülemeyecek sorunlara yol açabilir.

2. Performans Hızı

Performans hızı, kullanıcının işlemi yapabilme süresini ifade etmektedir. Yapılması gereken işlemin süresi yazılımın karmaşıklığıyla doğru orantılı olarak değişmektedir. Tasarımı iyi olan yazılımlar, kullanıcıya değişik engeller çıkararak hızlarını düşürürler. Kullanıcı kendisine sağlanması gereken bilgiyi yazılım içerisinde aramak zorunda bırakılır.

3. Kullanıcı Hatalarının Oranı

Yazılımı kullanma esnasında kullanıcı tarafında yapılan hata sayısı olarak ifade edilmektedir. Tüm kullanıcılar yazılım kullanırken bazı hatalar yapabilirler. Fakat iyi tasarlanmış bir yazılımda kullanıcıların hata yapması en aza indirilmelidir.

Kullanıcılar eğitsel materyal üzerinde çalışırken yeni bilgiler öğrenmektedirler. Bilgi eksikliği de öğrenme sürecinde hata yapmalarına sebep olabilir. Yazılım kullanımında yapılan bu hatalar öğrenme sürecini çok daha zor hale getirecektir.

4. Yazılım Kullanımını Hatırlama

Belirli bir süre sonrasında kullanıcıların yazılımın nasıl kullanıldığını ne kadar hatırlayabildiği önemlidir. Bu olgu, yazılım kullanımını hatırlama olarak ifade edilmektedir. Karmaşıklığı az olan yazılımlar, daha karmaşık olanlarına göre daha kolay hatırlanacaktır. Ayrıca benzer arayüz tasarımlarına yazılımların kullanımı, uzun bir süre sonrasında bile kolayca hatırlanabilecektir. Microsoft Office paket programlarını buna örnek verebiliriz.

5. Memnuniyet

Yazılım kullanılırken kullanıcı tarafından duyulan kullanım memnuniyetini ifade eder. İyi tasarlanmış, görsel olarak tatmin edici, kullanımı kolay yazılımlar kötü tasarlanmış ve kullanımı zor olan yazılımlara göre çok daha fazla beğenilip tercih edilmektedir. Tasarım rehberlerine sadık kalmak tasarımın kullanıcılar tarafından çok daha fazla sevilmesini sağlayacaktır. Eğer kullanıcılar yazılımda alışık ve güvenilir olmayan durumlarla karşılaşırsa yazılımı beğenmeyeceklerdir (Smith, 2015).

İnteraktif bir media sistemini en az yardımla kullanmak için, sistem içinde bulunan fonksiyonların ve bilginin yerleşimi ve bir bilgiye erişme yöntemi, görsel olarak açık ve net bir şekilde belirtilmelidir. Eğer arayüzün fonksiyonları ve bilgilerin hiyerarşisi, anlaşılır ve sade bir sıra içinde kurulmamışsa, kullanıcı sistemi kullanmayı öğrenirken bırakabilir. Sonunda da önerilen sistem ilk kullanımdan sonra terk edilebilir (Özcan, 2008).

Üstündağ'ın (1999) yaptığı çalışmada bilgisayar arayüz tasarımında grafik elemanlar ve internetin grafiksel etkileşim arayüzü üzerine çalışmıştır. Çalışmada arayüz tasarımındaki grafiksel elemanlar incelenmiş ve grafik arayüzün web sitesi tasarımındaki önemine dikkat çekilmiştir. Bilgisayar ortamını verimli kılmak üzere tasarlanmış grafiksel etkileşim arayüzlerinin

tipik görsel bileşenleri incelenmiş, web sayfalarında kullanılan görsel bileşenlerin evrensel olduğu ve grafik tasarım değerleri ışığında organize edilmesi gerektiği sonucuna varıldığı belirtilmiştir.

Ara yüz tasarımının amaçlarının gerçekleştirebilmesi için tasarım ilkelerine uyulması gerekmektedir. Eğer tasarım ilkeleri göz ardı edilirse yazılımın amaçlarına ulaşması beklenemez.

2.3.3 Öğretim yazılımlarında tasarım ilkeleri. Arayüzün amacı, kullanıcıya bir makine ile iletişimde olduğunu unutturarak yazılımla etkileşime geçmesini sağlamaktır.

Smith'e (2015) göre ara yüz tasarımının üç temel ilkesi şunlardır:

1. Hedef kitlenin tanınması;
2. Uygun tasarım yönergelerinin kullanılması;
3. Kullanıcıların hatırlanmasıdır.

2.3.3.1 Kullanıcıların tanınması. Tasarlanan ürünlerin kullanıcı tarafından kullanılacağı göz önüne alınarak kullanıcı kitlesinin özelliklerine ve kullanım çevresine uygun olarak tasarlanmalıdır (Maguire, 2001). Kaliteli ve kullanılabilir bir arayüzün sağlanmasında, hedef kullanıcı kitlesinin iyi analiz edilmesi ve tasarımların kullanıcı ihtiyaç ve beklentileriyle uyumlu olması gerekmektedir. Norman, (1988) kullanıcının bir sistemle ne yapacağını kolayca anlayabildiği ölçüde o sistemin kullanılabilir olabileceğini ifade etmiştir.

Shneiderman (1998) kullanıcı tanımlamasını kullanış profili ve işlem profili olmak üzere ikiye ayırmıştır. Kullanış profilinde, kullanıcının bireysel özellikleri ve farklılıkları belirlenirken, işlem profilinde ise, kullanıcının yazılımda gerçekleştirdiği etkinlikleri kolaylaştırma çalışmaları yapılmaktadır. Buna göre;

***Kullanış Profilleri*:** Kullanıcı profili çeşitliliğinin geniş bir şekilde ortaya koyulması gerekmektedir. Bu profillerin oluşturulmasında uygun olabildiğince çok etmene dikkat edilmesi gerekmektedir. Eğitimde öğrencilerin yaşı oldukça önemlidir. Hedef öğrencinin eğitim düzeyine göre tasarım farklılık gösterecektir. Diğer etmenlerden bir kaçı cinsiyet, eğitim seviyesi, fiziksel yeterlilikler, çalışma alanı ve etnik geçmiştir.

Tasarımcı kullanıcı grupla benzer özellikleri taşısa bile kullanıcı grubu temsil etmediğini unutmamalıdır.

Kullanış profillerinde diğer bir etmen de kullanıcıların öğrenme stilleridir. Bazı kişiler okuyarak bir bilgiyi çok iyi bir biçimde öğrenir. Bazı kişilerin ise ayrıca bilgiyi duyması gerekir. Bazı kişiler ise bilgiyi ilgili resimle ilişkilendirmek ister. Bu yüzden de tasarım değişik öğrenme stillerine olanak sunmalıdır.

Son olarak, kullanıcıların teknoloji konusunda uzmanlık düzeylerini dikkate almalıyız. Kullanıcıların bu yazılımı daha önce kullanıp kullanmadıkları ve bilgisayarı kullanma düzeyleri tespit edilmelidir. Eğer yazılımı veya bilgisayarı ilk kez kullanacaklarsa daha fazla yardıma ve rehberliğe ihtiyaçları olacaktır.

***İşlem Profilleri*:** Kullanıcıların yerine getirmeleri gereken işlemler belirtilmelidir. İşlem analizi farklı kullanıcı biçimleriyle işlemler listesinin eşleştirilmesi kadar basit bir değişkenler tablosu olabilir.

İşlem analizinin gereksinimlerin toplanması sürecinde yer alması gerektiği düşünülebilir. Yazılımın herhangi bir yerinde kullanıcının hareketlerini tayin etme tasarımcının işlemidir. En sık meydana gelen işlemlerin bilinmesi hangi ekranda hangi işlemlerin yerine getirileceğinin karar verilmesinde yardımcı olacaktır.

Ayrıca işlem analizi bize hangi işlemlerin basit, tek adımlık işlem olacağının belirlenmesinde yardımcı olacaktır. İşlemleri daha karmaşık ve alışılmamış hale getiren komutlar ekrandan çıkartılacaktır. Daha az fare ve klavye işlemi gerektiren komutlar tercih edilecektir.

Eğer yazılım geniş bir kullanıcı kitlesinde uygulanacaksa belirli bir kültürün kökeninde yer alan sembollerin ve mecazi ifadelerin kullanımında çok dikkatli olunmalıdır. Kültürel simgeler yerine uluslararası tanınmışlığı olan simgelerin ve sembollerin kullanımı tercih edilmelidir (Smith, 2015).

Kullanıcının özellikleri ve beklentileri belirlendikten sonra yazılımın uygun tasarım yönergelerine göre geliştirilmesi gerekmektedir.

2.3.3.2 Uygun tasarım yönergelerinin kullanımı. Tasarım alanında geliştirilmiş yönergeler bulunmaktadır. Bunlar grafik, ara yüz ve etkileşim tasarımı için başvuru kaynaklarıdır. Buna göre tasarımcı olarak;

- Tutarlı ve esnek olunmalı;
- Geri dönütler sağlanmalı;
- İşlemin kullanıcı tarafından gerçekleştirilmesi sağlanmalı;
- Kullanıcıların yaptığı hatalar giderilmeli;
- Kullanıcı sınırlılıkları farkına varılmalı;
- Kullanıcıların iç kontrol hattı desteklenmelidir (Shneiderman, 1998).

1. ***Tutarlılık ve Esneklik*****:** Tasarım elemanlarından benzer işlemi gerçekleştirenlerin benzer görünüme sahip olmasına dikkat edilmelidir. Ayrıca farklı işlemleri gerçekleştiren elemanlarının görünümleri birbirlerini andırmamalıdır. Tasarımları kullanıcıların daha önceden kullandıkları yazılımlara uygun olarak geliştirmeliyiz. Örneğin kullanıcılar menülerin

pencerenin üst bölümünde soldan listelenmesini beklerler. Ayrıca düzenlenebilir veya en azından okunabilir geniş beyaz okuma alanları bekleriz. Uzman kullanıcıların kullanmak isteyebilecekleri yerlerde kısayollar sağlanmalıdır. Etkinlikler, öğrenmeyi gerçekleştirmek için mümkün olduğunca fazla sağlanmalıdır. Böylece kullanıcılar kendilerini çok daha rahat hissedeceklerdir. Örneğin çoğu kelime işlemci programlarında kaydetme işlemini gerçekleştirmek için birçok yol sunulmaktadır. Klavye kısayollarından, menü elemanlarından ve araç çubuklarında yer alan düğmelerden kaydetme işlemi gerçekleştirilebilir. Bu işlemler aynı işlemi gerçekleştirmesine rağmen kullanıcılara kendilerine en uygun olanı seçme imkânı sunulmaktadır (Shneiderman, 1998).

2. ***Dönüt sağlama*:** Görsel dönütler kullanıcıların yazılımı kullanmalarında oldukça önemlidir. Örneğin bir ekrandan başka bir ekrana kullanıcı geçiş yaptığında bunun görsel öğelerle geri bildirilmesi oldukça yardımcı olacaktır. Yeni bir başlıkla veya vurgu düğmeleriyle kullanıcının geçiş yaptığı ekran belirtilebilir. Ayrıca fare ile üzerine gelindiğinde rengi değişen butonlar hem ilgi çekici hem de etkili olacaktır (Shneiderman, 1998).
3. ***İşlemi bitirmenin kullanıcı tarafından sağlanması için*:** Tanımlanmış işlemin sıralı adımlarla dizayn edilmesi gerekir. İşlemler gerçekleştiğinde bilgilendirme sağlanmalıdır. Böylece bu etkinlik kişinin zihinsel listesinden çıkarılacaktır. İşlemi gerçekleştirme kişiye başarı duygusu hissi kazandırır. Yeni bir işleme kişinin hazırlanmasını sağlar. "Tamam" ve "iptal" butonlarıyla etiketlendirilmiş diyalog kutularıyla kişiler yapılması gereken işlemi gerçekleştirdiklerini güçlü bir şekilde hissedeceklerdir (Shneiderman, 1998).
4. ***Hataları giderme*:** Yazılımlarda bulunması gereken özelliklerden bir diğeridir. Eğer kullanıcı bir hata yaparsa, yazılım bu hatayı telafi ederek düzeltmelidir. Örneğin "kaydet" ve "sil" butonları bir yazılımda yan yana yerleştirildiğinde kullanıcı kaydet yerine yanlışlıkla sil butonuna basmış olabilir. Bu durumda yazılımın kullanıcıdan onay isteyerek bilgilendirmesi

hatalı kullanımları engelleyecektir. Ayrıca uzun video ve animasyon gösterimlerinde kullanıcıdan onay alınması gerekir. Meydana gelebilecek hataların planlanması gerekir. Örneğin verilerin parantez içerisinde girilmesi gerektiğinde eğer kullanıcı tam tersini yaparsa yazılımın bu parantezleri tamamlaması beklenir. Yazılımda yedekleme ve başka bir yere geçiş olanağı olmalıdır. Kullanıcı istediği zaman yazılımdan çıkabilmelidir. Kullanıcı bir hatayı düzeltmek için yazılımdan çıkmak zorunda bırakılmamalıdır (Shneiderman, 1998).

5. ***Yazılımda kullanıcı sınırlılıklarının farkına varma*:** Bilgiler uygun biçimde gruplandırılmalıdır. Uzun menü listelerini hatırlamak kullanıcılar açısından oldukça zordur. Bunun için benzer öğeler aynı menü başlığında toplanmalıdır. Böylece kullanıcı menü öğesini aramak için boş yere zaman harcamaz. Kullanıcıya yardımcı olmak için;
   - Bellek yükü azaltılmalıdır. Eğer yazılım kullanıcı yerine bazı işlemleri gerçekleştirebiliyorsa, bunları yazılımın yapması sağlanmalıdır. Örneğin bir hesap işlemi yazılımın kullanımında gerçekleştirilecekse hesaplama işlemlerini yazılım yapmalıdır. Kişi başka bir programı çalıştırmaya gerek duymamalıdır.
   - Butonların ve simgelerin kullanım amaçları açık bir şekilde belli olmalıdır. Yazılımda yer alan butonların işlemini anlamak için bunlara basmak boşuna zaman kaybıdır. Ayrıca bir de butona basıldığında istenmeyen bir işlem gerçekleştirilirse durum çok daha kötü olacaktır. Aynı özellik araç çubukları ve menü elemanları için de geçerlidir.
   - Yazılımda tüm özellikler açık olarak geliştirilmelidir. Eğer bir nesne tıklanabilir özellikte olacaksa görünümü ile bunu kullanıcıya gösterebilmelidir. Ayrıca kullanıcıların bir işlemi gerçekleştirebilmeleri için bazı becerilere ve bilgilere ihtiyaç duyacaklarsa bu bilgiler işleme başlamadan önce kullanıcıya bildirilmelidir.

- Yazılım içerisinde kullanıcıyı boşa çıkaran engeller ve fazlalıklar ortadan kaldırılmalıdır. Eğer kullanıcı kayıt işlemi yapmak istediğinde sürekli olarak kayıt için onay düğmesi ekrana çıkartılırsa kullanıcıya işlem akışı engellenmiş olur. Farenin her tuşuna bastığımızda bileklerde gerilme meydana geldiğini unutmayınız. Bunun için benzer işlemleri gerçekleştirmek için yapılması gereken işlem sayısı azaltılmalıdır.
- Gereksiz kontrol işlemlerinden kaçınılmalıdır. Her ekranda yardım menüsüne ulaşmak için kullanıcıya çok fazla sayıda yol sunulursa kullanıcı yardım menüsüne nasıl doğru biçimde ulaşacağı konusunda şüpheye düşebilir. Bu yüzden gereken durumlar dışında aynı işlemi gerçekleştirmek için farklı yolların sunulması çok yararlı olmayacaktır.
- Yazılımı kullanırken yapılması gereken işlemler basitleştirilmelidir. Yazılımı kullanırken kullanıcıların nelere ihtiyaçları olacakları ve yazılımdan neler bekleyebilecekleri tanımlanmalıdır. Arayüzle ilgili her kararda kullanıcının işlemleri kolaylaştırmak temel noktamız olmalıdır. Eğer bir butona yazılımın neredeyse tüm aşamasında ihtiyaç duyacaksak, butonu her ekran aşamasında aynı yerde bulundurmalıyız. Eğer yazılım içerisinde geçiş işlemleri fazla yapılacaksa geçiş işlemlerin iyi tasarlanması gerekecektir (Shneiderman, 1998).

6. ***Yazılım içerisinde kontrolün desteklenmesi*:** Kullanıcılar; özellikle de uzman kullanıcılar, yazılımın kendi kontrolünde olduklarını hissetmek isterler. Kendi yaptıkları işlemlere yazılımın yanıt vermesini isterler. Eğer yazılım kullanıcının kendi başlattırmadığı bir komutu çalıştırırsa, kullanıcı kendisini rahatsız edecektir. Ayrıca kullanıcı yazılım içerisinde ne yapmasını istediğini bilmesine rağmen istediklerini yazılımda yerine getiremezse kullanıcı hayal kırıklığına uğrayacaktır. Bu tür durumlar kullanıcının memnuniyetsiz olmasına ve endişe duymasına yol açacaktır (Shneiderman, 1998).

2.3.3.3 Kullanıcıların hatırlanması. Tasarım sürecinde yapılan her şey, kullanıcı ihtiyaçları üzerine yapılandırılmalıdır. Yazılımdaki tasarım özellikleri bizim ihtiyaçlarımıza göre değil, yazılımı kullanmak için bilgisayarın karşısına geçecek insanların ihtiyaçlarına göre düzenlenmelidir. Kullanıcılar belirli işlemleri gerçekleştirmek isterler. Bu işlemleri eğer biz zorlaştırırsak kullanıcılar yazılımı kullanmak istemeyeceklerdir.

IBM yazılım geliştirme grubuna göre;

- **Ticari Amaçların Oluşturulması:** Tüm tasarım ve kullanıcı katılımlarında temel odak noktası hedef pazarın belirlenmesidir.
- **Kullanıcıların Anlaşılması:** Eğer kullanıcıların yazılımınızı anlamalarım istiyorsak, ilk önce siz kullanıcılarınızı anlamalı ve onları tanımalısınız.
- **Rekabetin Takdir Edilmesi:** Üstün bir tasarım devamlı olarak kullanıcılarının ve rekabetin farkında olmayı gerektirmektedir. Kullanıcının gereksinim duyduğu işlemleri gerçekleştiren alternatiflerle kendi yazılımızın sonuçları karşılaştırılmalıdır.
- **Tüm Kullanıcı Tecrübelerinin Dikkate Alınması:** Kullanıcının yazılımda duyularıyla etkileşimde bulunduğu her şey tasarım ekibi tarafından dikkatlice geliştirilmelidir. Ürünün reklamı, sipariş verilmesi, alımı, paketlenmesi, korunumu, kullanımı, yönetimi, dokümantasyonu, güncellenmesi, desteklenmesi dikkate alınması gereken özelliklerdendir.
- **Tasarımların Değerlendirilmesi:** Kullanıcı geribildirimleri ne kadar sık ve erken toplanırsa, ürün daha iyi gelişir ve ihtiyaçları karşılar.
- **Sürekli Olarak Kullanıcı Gözlemlerinin Yönetimde Kullanılması:** Ürün kullanıldığı sürece kullanıcılar gözlemlenmeli, dinlenmelidirler. Bu bilgilerden yararlanılarak pazar alanında değişiklikler yapılıp, rekabet arttırılabilir (IBM, 2015).

2.3.4 Öğretim yazılımlarında ekran tasarım özellikleri. Bilgisayar teknolojisinin hızla gelişmesinin eğitim sistemine de etkileri olmuştur. Bu alanda birtakım teknolojik yeniliklerin getirilmesi kaçınılmaz hale gelmiştir. Öğrenme ortamında yeni teknolojilerin (bilgisayar, video, televizyon, uydu vb.) kullanılması öğrenmeyi olumlu yönde etkilemektedir (Merill, 1992). Eğitim ortamında bireylerin ihtiyacının teknolojik imkânlarla karşılanması onu kullanma zorunluluğunu beraberinde getirmiştir.

Bilgisayar destekli eğitimde en az üç boyut bulunmaktadır. Birincisi, eğitim-öğretim faaliyetinde kontrol rolünde olan öğretmendir. İkincisi, öğrenme karşılamak için tasarlanmış yazılımların çalıştırılabileceği bilgisayar donanımıdır (Merill, 1992; Pelgrum & Plump, 1991). Üçüncüsü ise öğrenci ile makine arasında etkileşim sağlayan öğretim yazılımlarıdır (Şeniş, 1991).

Bilgisayar destekli öğretimde kullanılacak ders yazılımlarının ekran tasarımının önemli bir hale gelmiştir. Çünkü yazılımın hazırlanmasından çok o yazılımda eğitsel özelliklerin olması, öğrenciyi güdülemesi ve içeriğin anlaşılır olması beklenmektedir. Örnek olarak eğitici oyunlar, renkler, yazı stilleri, grafikler vb. gibi faktörler öğrenciyi etkin bir öğrenmeye katması bakımından oldukça önemli özelliklerdir (Bülbül, 1999).

Seçilecek ekran özelliklerinin neler olduğunun iyi bilinmesi gerekmektedir. Ekrandaki grafiksel metinin tek başına bir anlamı olmayıp aynı zamanda geçmiş edinilmiş bilgilerle etkileşime de geçmektedir. Öğretim yazılımları tasarlanırken yazılım ekibinin dikkat etmesi gereken alan uzmanlarının da üzerinde görüş birliği ettikleri bazı standartların olması şarttır. Buna göre, Bülbül (1999) ekran tasarım standartlarını metin düzeni, yerleştirme, görünüm ve grafik başlıkları altında gruplandırmıştır:

2.3.4.1 Metin Düzeni

Etkili bir eğitim yazılımı tasarlarken metin düzeninde aşağıda belirtilen kurallara uyulması gerekmektedir.

- Metinleri oluşturan paragraflar ekranda kolaylıkla görülebilen ve okunabilecek bir biçimde yer almış olmalıdır.
- Cümleler kısa ve anlamlı bir biçimde sunulmalıdır.
- Satır sonlarında kelimeler anlam bozmaması için bölünmemelidir.
- Paragraflar bölünmeden aynı ekranda bitirilmelidir.
- Aynı ekranın farklı yerlerinde ilgisi olmayan bir dikkat çekici kullanmaktan kaçınılmalıdır.
- Metin ifadelerinde yazım kurallarına uyulmalıdır.
- Yönerge cümleleri olumlu yapıda olmalı çok sayıda teknik kelime ve kısaltmalar kullanmaktan kaçınılmalıdır.
- Her paragraf için o paragrafı özetleyecek bir başlık kullanılmalıdır.
- Başlıklar üç satırı aşmamalıdır.

2.3.4.2 Yerleştirme

Etkili eğitim yazılımı tasarlarken görsel öğelerin yerleşiminde aşağıda belirtilen kurallara uyulması gerekmektedir.

- Öncelikle görülmesi gereken ifadeler ekranda verilmelidir.
- Paragrafların kolay okunması için uygun satır aralıkları verilmelidir.
- Gerektiğinde farklı erişimler için yapılabilecek işlemlerle ilgili yönergeler bulunmalıdır.
- Ekranda istenildiğinde destekleyici bir bilginin gözükebilmesi için yer ayırılmalıdır.
- Paragraflar arasında en az bir satır boşluk olmalıdır.

2.3.4.3 Görünüm

Yapılan araştırmalar, öğrencilerin kırmızı renkte gördüklerini daha uzun süre zihinlerinde sakladıklarını ortaya koymuştur. Dolayısıyla bilhassa hatırlanması istenen materyaller için kırmızı renk kullanılabilir. Diğer yandan mavi öğeler dikkati en az çeken öğelerdir. Daha az öneme sahip öğelerde mavi renk kullanılabilir. İnsanlar ilk önce sarı nesnelere bakma yatkındırlar. Bir görselde önemli öğeler ya da anahtar kelimeler sarı renk kullanılarak öne çıkarılabilir. Ayrıca, okunabilirlik açısından en iyi renk birleşimi sarı zemin üzeri siyah yazıdır. Bilgisayar destekli öğretimde bilgi işlem sürecini kolaylaştırmak için yazılı bilgiler renk yanında, koyu yazı, altını çizme, kutu içine alma gibi yardımcı araçlarla desteklenmelidir. Bu teknikler önemli bilgileri ön plana çıkarmak ve öğrenenlerin ilgilerini çekmek için kullanılabilir. Ancak bu araçların kullanımında tutarlı olunmalıdır. Yani, araçlar bir yerde hangi amaç için kullanılmışlarsa başka yerlerde de aynı amaç için kullanılmalıdırlar. Öte yandan çok önemli mesajları vurgulamak için yanıp sönme ya da parlama gibi teknikler kullanılabilir, ancak bu teknikleri aynı ekranın iki farklı yerinde kullanmaktan kaçınılmalıdır (Yalın, 2001).

Bülbül (1999) görünüm konusunda şunları belirtmiştir:

- Programın ilerleme, geri gitme, yardım, çıkış vb. yönergelerle ilgili buton işaretleri kullanıcının kolayca görebileceği bir yerde ve nitelikte olmalıdır. Farklı ekranlarda dahi bu yönerge düğmeleri aynı pozisyonda olmalıdır.
- Ekrandaki öğelerin hareketlerinin gözü yormamasına dikkat edilmelidir.
- Kullanılan yazı tipi öğrenci düzeyine uygun olmalıdır.
- Konu ile ilgili kullanılan renk ve grafikler öğrencinin ilgisini çekecek nitelikte olmalıdır.
- Kullanılacak renklerin tercihinde "Renk Bilgisi"nden yararlanılarak hareket edilmelidir.

- Dikkat çekilmek istenen kavramlar, farklı yazı çeşidi veya farklı renk kullanılarak vurgulanmalıdır.
- Paragraf başı konunun önemini belirtmek için içerden başlamalıdır.
- Bilginin verilmesinde "Küçük adımlar " ve "Aşamalılık" ilkelerine uyulmalıdır.
- Yeni bir sayfaya ancak öğrencinin onayı ile geçilebiliyor olmalıdır. Aksi takdirde ekran onay beklemelidir.
- Ekranda dört farklı renkten fazla renk kullanılmamalıdır.

2.3.4.4 Grafik

Grafik, animasyon, diyagram, çizelge, harita ve resim gibi sözsüz görsel materyaller ve koyu yazı, altını çizme, kutu içine alma gibi yardımcı araçların tasarımı ve kullanımı öğrenme üzerinde önemli bir etkiye sahiptir. Görsel bilgiler sözel bilgileri tanımlamak, açıklığa kavuşturmak veya desteklemek amacıyla kullanılır. Birçok araştırma görsel mesajların öğrenme üzerinde etkili olduğunu ortaya koymakla birlikte, bu mesajların öğretimde etkililiği, yerinde ve uygun kullanımına bağlıdır. Görsel materyallerin etkililiği aynı zamanda, bunların bilginin özünü sunmalarına, önemli noktaları ortaya koymalarına, yazılı materyalde verilen yapısal ilişkileri açıklamalarına ve özellikle bilgisayar destekli öğretimde ilgili yazılı materyalle yan yana verilmelerine bağlıdır. Aksi takdirde bu materyallerin kullanımı öğrenme üzerinde olumlu bir etki yaratmadığı gibi, kavramayı zorlaştırabilir. Ayrıca, eğer yazılı materyal iyi organize edilmiş veya basit ise görsel materyaller öğrenme üzerinde fazla bir etkiye sahip olmayabilir. Grafiklerin kullanımında dikkat edilmesi gereken noktalar şunlardır:

- Grafiksel bilgi, diğer öğretimsel mesajlarla tutarlı ve onlarla bütünleştirilmiş olmalıdır.

- Görsel materyallerde aşırı ayrıntı ya da gerçek resimleri kullanmaktan kaçınılmalıdır. Basit çizimler, ana fikri gerçek resimlerden daha açık gösterebilir.
- Karmaşık grafikler mümkün olduğunca basit parçalara bölünerek verilmelidir. Parçalar aynı ekran üzerinde aşamalı olarak üst üste yerleştirilerek basitten karmaşığa doğru verilebilir.
- Öğrenen, sunulan grafiğin izleme süresini kontrol edebilmelidir.
- Canlandırmalar gerektikçe tekrar edilebilmelidirler.
- Grafikler ilgili yazılı bilgilerle aynı ekranda verilmelidir. Böylece öğrenen grafik ve onunla ilgili açıklamayı birlikte öğrenebilir. Ancak, karışıklığa sebep olmaması ve öğrencilerin dikkatlerini çekmesi için aynı ekranda metinle birlikte kullanılan grafikler kutu içine alınmalıdır.
- Önemli unsurlara dikkat çekmek ya da unsurları birbirinden ayırmak için renk kullanılmalıdır. Ancak bir ekrandaki renk sayısı en fazla dört renkle sınırlandırılmalıdır (Yalın, 2001).

Bülbül (1999) grafik konusunda dikkat edilmesi gereken noktaları şu şekilde ifade etmektedir:

- Verilen metin gereken görsel (grafik ya da resim) metinle birlikte aynı ekranda yer almalıdır.
- Kullanılan grafik ya da resimlerde gereksiz ayrıntı bulunmamalıdır.
- Konuların anlatılmasında gerekliyse canlandırılmış resim (animasyon) kullanılmalıdır**.**

İpek, (2001) genel olarak ekran tasarımı ilkeleri ve kuralları şu şekilde özetlemektedir:

***Yazı stili*:** Grafik düzenlemede, serif ve sans-serif, Chicago, times, yazı tipi (font) iyi görünmekle beraber, palotini ve helvetica daha kullanışlı görünmektedir. Bu fontlarda koyu görünüş, altı çizili kelimelerden daha iyi sonuç vermektedir.

***Küçük ve büyük harfleri kullanma*:** Bilgisayar ekranında küçük harflerin okunuşu büyük harflerden kolaydır.

***Kelimelerin veya cümlenin altının çizilmesi*:** Yararlı olan bu yöntem yazılı metinin ortasında kullanılırsa sonraki satırların okunuşu zorlaşır. Daha sonraki cümleler italik yazılırsa veya koyu ve renklerin değişik tonlarında değişimi şekilde ise bu seçeneğe alternatif oluşturulabilir.

***Parlama ve sönme (flushing)*:** Çok özel durumlarda kullanılabilir. Bu kullanım okuyucunun gözlerini rahatsız edebilir. Bu yüzden ekranda iki ayrı yerde kullanılmamalıdır.

***İtalik yazma*:** Kitaplarda çok kullanılmakla birlikte ekranda fazla kullanılmaması gerekir. Ekranda okuma güçlüğü meydana getirmektedir.

***Renklerin kullanılması*:** Renkler bilgisayar ekranındaki konuları açıklamak ve görünür duruma getirmek için kullanılmakta, fakat renklerin gereksiz ve bu amaca aykırı kullanıldığı da görülmektedir. Bilgisayarlar çok fazla sayıda renklere sahip olmakla birlikte, en etkili renkler siyah ve beyaz olmaktadır. Özellikle çok sıcak renkler olan mor ve pembe renkten kaçınmak gerekir, yeşil ve mavi renkler daha çok tercih edilmelidir. Siyah arka plan üzerinde beyaz, sarı, koyu mavi ve yeşil daha iyi okunan, mor ve kırmızı ve mavi daha az okunabilen renklerdir.

***Satır uzunluğu*:** Kısa satırları okumak kolay olurken; uzun satırlar göz hareketi nedeni ile cümlenin başlangıcı ve sonundaki kelimeler arasındaki hareketi zorlaştırır. Her satırda 8-10 kelime bilgisayar ekranı için uygundur. Bu yaklaşık olarak 80 harfi içerebilir. Kısa cümleler daha değişik ve değişkenliğe sahip sayfa tasarımı olanağı verir.

***Yazıların yoğunluğu*:** Basılı yayınlar için her sayfada kırk satır ve her satıra 60 harf ile 400 civarında kelime uygundur. Ekranda ise 18 satır, her satırda 39 harf ve her sayfa için 120 kelime uygundur.

***Harf büyüklüğü*:** Okuyucuların 60 cm uzaktan okumaları halinde 12 harf (font size) büyüklüğü uygun görünmektedir.

***Ekran/sayfa kenarlarındaki boşluklar*:** Ekran tasarımında uluslararası ve kültürel etkenler önemlidir. Yazıların soldan tasarımı okumayı kolaylaştırır. Kelimeler ve cümleler arasındaki sıklaşma ile göz hareketi dağılmaz. Bu da okumayı daha da kolaylaştırır.

Rızvanoğlu'da (2009) çocuk kullanıcılar için arayüz tasarımında dikkat edilecek hususları şu şekilde sıralamıştır:

- Bazı çocuklar daha erken yaşlarda okumaya başlasa da, özellikle 2-7 yaş grubu çocuklara yönelik tasarımlarda bu yaş grubundaki çocukların halen okuyamadığını göz önünde bulundurmak gerekmektedir. Bu gruptaki çocuk kullanıcılara yönelik tasarlanan arayüzlerde, metin yerine ses, görüntü ve animasyon formatlı içeriklerin kullanılması tercih edilmelidir. Ancak, çocukların bir kere de kavrayabilecekleri unsur sayısı gelişmekte olan bilişsel becerileriyle sınırlı olduğundan, renkli ve hareketli içeriğin aşırı kullanımından kaçınmak gerekmektedir.
- Okuma yazma bilen daha üst yaşlardaki çocuklara yönelik arayüz tasarımında yoğun, süslü ve hareketli metin içerikleri ve talimatlar kullanmaktan kaçınılmalıdır. Kullanılan metin içeriklerinin, kısa, basit ve kolay kavranabilir yapıda olmasına dikkat etmek gerekmektedir. Sadece metin tabanlı bağlantıların kullanımından kaçınılmalıdır.
- Arayüzlerde metin içeriği kullanılırken, font büyüklüğünün tespitinde, yaş ve font büyüklüğü arasında ters bir orantının benimsenmesi önerilmektedir. Yaş küçüldükçe kullanılan font büyüklüğünün artırılması olumlu olacaktır. 9-11 yaş aralığındaki çocuklar için arayüzlerde 14 punto font büyüklüğünü tercih etmek olumlu olacaktır.

- Fitts Kanunu'na göre, ekranda hedef ne kadar uzakta ve küçükse, ona fareyle tıklamak daha da zorlaşmaktadır (Hutchinson, Druin & Bederson, 2007). Dolayısıyla arayüzlerde, tıklanacak hedeflerin mümkün olduğunca büyük ve kolay tıklanabilir olması gerekmektedir.
- Arayüzlerde dolaşımı desteklemek üzere tanıdık, tutarlı, açık ve kolay kavranabilir metaforlar kullanmak gereklidir. Kesin bir çözüm olmamakla birlikte, arayüzlerde kullanılan üç boyutlu uzamsal metaforlar çocuk kullanıcılar tarafından olumlu karşılanmaktadır.
- Arayüzlerde dolaşım esnasında ana menüye erişim kolay olmalıdır.
- Arayüzlerde, özellik durum bilgisi ve hata bildirimi gibi çıktı unsurlar olarak görsel ve sesli uyarılar kullanılmalıdır. Ancak bu noktada, kullanılan unsurların aşırı ilgi çekici olmaması gerekmektedir. Aksi halde söz konusu unsurlar çocukların ilgisini gereksiz yere çekmekte ve sırf bu uyarıları harekete geçirmek için, çocuk kullanıcılar ısrarla aynı eylemi gerçekleştirmektedir.
- "Yardım" unsurlarına yazılı ve görsel olarak kolay ve hızlı erişim sağlamak gerekmektedir.
- Kaydırma çubuklarının kullanımından kaçınmak gereklidir.
- Çocuk kullanıcıların arayüzlerde bilgiyi arama ve bilgiye erişim süreçlerini desteklemek üzere, anahtar kelime merkezli arama yerine ikonlar aracılığıyla yapılabilen kategori merkezli arama seçeneğini sunmak daha faydalı olacaktır (Rızvanoğlu, 2009).

2.4. Programlama Öğretimi

Bu kısımda programlama konusu etraflıca ele alınmış, programlama öğretiminin; tanımı, tarihçesi, kullanılan diller, önemi, ülkemiz ve dünyadaki öğretimi, yanılgıları, yaşanan zorlukları, çeşitli öğretim yöntemleri, kullanılabilecek yazılımlar, bu yazılımların sınıflandırılması ve bu çalışmanın konusu ile ilgili yapmış araştırmalar üzerinde durulacaktır.

2.4.1 Programlama. Programlama kavramı için farklı tanımlamalar mevcuttur. Bilgiustam web sayfasına (2015) göre bilgisayara ya da elektronik devrelere yaptırılması gereken işi, bir dizi komutlar kullanarak yazmaktır. Vikipedi Özgür Ansiklopedi'ye (2015a) göre programlama, bilgisayarın donanıma nasıl davranması gerektiğini anlatan, bilgisayarı yönlendiren komutlar, kelimeler, aritmetik bir dizi işlemlerdir. Bir başka tanımda ise programlama, bilgisayar programlarının yazılması, test edilmesi ve bakımının yapılması sürecine verilen isimdir. Çölkesen'e (2002) göre ise programlama, gerçek hayatın modellenmesidir. Bilgisayar yazılımları bu modellerin temel alınarak gerçek hayat durumlarının bilgisayar ortamına aktarılmasıdır.

2.4.2 Programlama dilleri. Bilgisayarlar yapılacak işlemlerin onun anlayacağı dilde (makine dili) açıkça yazılmasını beklerler. Ancak bu işlemlerin zor ve karmaşık olan makine dilinde bir programcı tarafından yapılması yerine programlamanın yapısal biçimde bir dil sayesinde oluşturulması ve daha sonra tekrar makine diline çevrilerek bilgisayara verilmesi işlemini programlama dilleri üstlenirler. Programlama dilleri, programcının bilgisayara hangi verileri kullanacağını, verinin nasıl saklanıp iletileceğini, hangi koşullarda ne tür işlemlerin yapılacağını tam olarak anlatmasını sağlar (Vikipedi, 2015b).

Eryılmaz'a (2003) göre programlama dili, programcının bilgisayarın ne yapmasını istediğini anlatmak için tasarlanan yapay bir dildir. Programlama dilleri ile programcı bilgisayara hangi verileri ne şekilde kullanması gerektiğini ve hangi sonuçları vermesini istediğini iletir. Bilgisayarlarla kullanıcı arasındaki iletişimi sağlayacak bir mekanizmaya ihtiyaç duyulmuştur. Başlarda bu iletişimi sağlamak için makine dili (Assembly) kullanılmıştır. Assembly dilinin öğrenmesi ve kullanılmasının zor olması programlama dillerinin doğmasına sebep olmuştur. 1957 yılından itibaren Fortran, Algol, Cobol, Basic, Pascal, C gibi programlama dilleri

geliştirilmiştir. Özellikle 1980'lerden sonra bu programlama dillerinin daha gelişmiş versiyonları ve nesneye yönelik programlamayı destekleyen Delphi, Java, C++, C#, Visual Basic gibi üst düzey programlama dilleri geliştirilmiş ve programların oluşturulmasında bu dillerin kullanımı ağırlık kazanmıştır.

Herhangi bir programlama dilinde yazılan kaynak kodlar genellikle bir derleyici (complier) ve yorumlayıcı (interpreter) sayesinde belirli bir sistemde çalıştırılır hale getirilir. Ayrıca kaynak kodu, bir yorumlayıcı aracılığıyla derlemeye gerek kalmadan da satır satır çalıştırılabilir. Derleyici, yazılan programları tarayıp içerisinde mantıksal veya yazınsal (syntax) hataları tespit eden, bulduğu hataları kullanıcıya gösterip programın düzeltilmesine imkân tanıyan, eğer hata yoksa programı çalıştırıp sonucunu gösteren, ayrıca türüne göre pek çok başka özelliği barındırabilen birer platformdur. Yorumlayıcı ise yazılımı parça parça ele alarak doğrudan çalıştırır (Vikipedi, 2015a). Yorumlayıcılar çalıştırılabilir kod üretmezler. Yorumlama süreci aşama aşama yapılmadığı için genellikle ilk hatanın bulunduğu noktada programın çalıştırılması bırakılır. Derleyicilerin aksine kodun işlenmeyen satırları üzerinden hiç geçilmez ve buralardaki hatalar ile ilgilenilmez. Yorumlayıcılar genelde kaynak koddan, makine diline anlık olarak çevirim yaptıkları için, derleyicilere göre daha yavaş çalışırlar. Ayrıca kodu iyileştirme (optimizasyon) imkanı da çoğu zaman yoktur (Vikipedi, 2015c).

2.4.3 Programlama dili becerileri. Bilgisayar programlama dilleri, içerisinde üst düzey bilişsel becerilerin yer alması nedeniyle öğretimi ve öğrenimi zor bir alandır. Programlama dillerinin zorluğu, aynı zamanda öğrenme içeriğinin geniş olmasıyla da açıklanabilir. Bu alanın içeriğini kısaca özetlemek gerekirse;

- Bilgisayar okur-yazarlık bilgisi;
- Donanım bilgisi;

- Programlamaya ilişkin temel kavramlar bilgisi;
- Programlama dilinin “dil yapısı” bilgisi;
- Problem çözme becerisi.

Yukarıdaki maddelerin kısa açıklamaları aşağıda verilmiştir:

- **Bilgisayar okur-yazarlık bilgisi:** Programlama dillerinin amacı, herhangi bir platformda çalışabilecek programlar oluşturmaktır. Bu nedenle programlama öğretebilmek için öğrenenlerde öncelikle işletim sistemine ait temel bilgilerin olması/verilmesi gerekmektedir. Ancak bu bilgi programlama dilleri dersi kapsamından ziyade bu derse ilgili ön bir öğrenme şeklinde ele alınabilir.
- **Donanım bilgisi:** Programlama dillerinin her bir komutu, bilgisayarın bir donanımını kullanıp ona iş yaptırmaya yöneliktir. Bu sebeple, öğrenenlerin bilgisayarın donanım bilgisine sahip olması gerekir. Bu bilgi de, bir önceki bilgisayar okur-yazarlık bilgisinde olduğu gibi ön öğrenme şeklinde verilmelidir.
- **Programlamaya ilişkin temel kavramlar bilgisi:** Programlama becerilerindeki belki de en önemli bilgi olarak temel kavram bilgisi ortaya çıkmaktadır. Günümüzde informal programlama dilleri eğitimine sahip birçok kişi “komut” kavramını bilmeden komutlar yazmakta, “değişken” kavramını anlamadan değişkenleri kullanmaktadır. Programlama dilleri bu şekilde birçok kavramı içerisinde bulunduran bir alan olup formal bir eğitim için bu alanda ön öğrenme olarak temel kavramlar ile başlanmalıdır.
- **Programlama dilinin “dil yapısı” bilgisi:** Programlama dillerinin anlamlı en küçük parçasına “atom” (token) denir. Atomlar bir araya gelerek sözcükleri (lexical), sözcükler kurallı bir şekilde bir araya gelerek söz dizilimini (syntax) oluştururlar. Söz dizilimleri anlamlı olduğu sürece bilgisayara bir iş yaptırabildikleri için aynı zamanda anlamlı (semantic) olmak

zorundadır. Günümüzde sözcüksel ve söz dizilimsel olarak birbirinden farklılık gösteren çok sayıda programlama dili geliştirilmiş olmasına karşın tüm bu dillerin yazım kurallarını tanımlamak için oluşturulan yazım kuralları başvuru kaynağı olan Backus-Naur gramatik yapısı kullanılır.

- **Problem çözme becerisi:** Programlar diğer adıyla yazılımlar, komutlardan (kodların) meydana gelmiştir. Ancak komutların sıralanışı rastgele değil, tam tersine bir problemi çözmek üzere planlı şekilde tasarlanmalıdır. Bu nedenle, öğrenenlerin bir problemi çözmek için bir "program tasarımı" yapmaları gerekmektedir. Bu durum ise problem çözme becerisi ile açıklanabilir (Gültekin, 2006).

2.4.4 Programlama öğretimi. Programlama öğretimi geçen yıllara rağmen çok fazla değişime uğramadı. Genellikle öğretmenler, yeni kontrol yapılarını veya veri yapılarını sunar, öğrencilere bir kaç örnek gösterir ve onlardan karşılaştıkları problemi çözmelerini beklerler. Bu konuda öğrencilerin yakındıkları konu, hep öğretmenin anlattıklarını izleyip anladıkları fakat kendi problemlerine çözüm geliştiremedikleri olmuştur (Garner, 2003).

Programlama öğretimine yönelik araştırmaların çoğu öğrencilerin programlamaya giriş dersleri boyunca kazandıkları programlama bilgisi ve programlama öğrenirken öğrencilerin gerçekleştirdikleri bilişsel işlemlerle ilgilenmişlerdir. Buna göre programlama öğrenimi süresince birbirleriyle ilişkili üç tip programlama bilgisinden söz edilir (Bayman & Mayer, 1988):

1. **Yazımsal (Syntactic) Bilgi:** Belirli bir programlama diline ait kullanım kurallarının bilgisidir. Yazımsal bilgi dil özellikleri veya gramer bilgisi olarak tanımlanır. Örnek olarak Pascal programlama dilinde "Repeat-Until" döngüsünün yazımı veya her komutun sonuna noktalı virgül konulması gibi durumlar ele alınırsa; bu teknik bilgi derlenecek olan programın yazımı için gereklidir, fakat bir probleme çözüm üretmek ya da tasarım yapmak için yeterli değildir.

2. **Kavramsal (Conceptual) Bilgi:** Programlama kavramlarının (döngü, şartlı döngü, koşul yapıları gibi) ve prensiplerinin bilgisidir. Yazımsal bilgi bir dilin yapısal (gramer) kurallarının bilgisi olurken kavramsal bilgi, bir probleme çözüm üretebilmek için bu yapıların mantığının ve ne iş yaptıklarının tam olarak anlamadır diyebiliriz.
3. **Problem Çözme – Stratejik (Strategic) Bilgi:** Yazımsal ve kavramsal bilgileri kullanarak bir probleme çözüm sunabilme kabiliyetidir. Stratejik bilgi karşılaşılan problemi fark etme, anlama, tanımlama ve çözümü için alternatif yollar geliştirebilmedir.

Bayman ve Mayer (1988) tarafından yapılan deneysel çalışmada, kavramsal ve stratejik bilginin ilişkili olup olmadığı denenmiştir. Bu çalışma, BASIC programlama dilini öğrenen ve programcılığa yeni başlayan bireyler üzerinde yapılmıştır. Bu çalışmada, kavramsal bilgi ile problem çözme performansı (stratejik bilgi) birbirleriyle çok güçlü bir ilişki içerisinde olduğu sonucu çıkmıştır.

McGill ve Volet (1997), Bayman ve Mayer'in yukarıda verdiği programlama bilgilerini öğretim sürecini Tablo 2.1'de şematikleştirmiştir.

Tablo 2.1

Programlama Bilgilerinin Değişik Boyutları

	Bildirimsel Bilgi (Declarative)	İşlemsel Bilgi (Procedural)
Yazımsal Bilgi (Syntactical)	Programlama dilinin yazım kuralları bilgisi. Örnek: Java programlama dilinde her komutun sonuna noktalı virgül işaretinin konulması gerektiğini bilme.	Program yazarken yazım kurallarına uyabilme. Örnek: Java programlama dilindeki While komutunu yazım kuralı olarak doğru yazabilme
Kavramsal Bilgi (Conceptual)	Program çalıştırıldığında hangi mantıksal işlemlerin yapıldığını anlama ve açıklama becerisi. Örnek: Yalancı (Pseudo) kodun ne iş yaptığını açıklayabilme.	Bir programlama problemine yönelik çözüm oluşturabilme. Örnek: Bazı verilerin ortalamalarını bulan bir "Procedure" dizayn edebilme.
Stratejik Bilgi (Problem Solving)	Sıra dışı bir problemle karşılaşıldığında buna ait bir çözüm üretmek için program tasarım, kodlama ve test edebilme becerisi.	

Tablo 2.1'deki bildirimsel bilgi, öğrencilerin bir programlama dilinin genel kurallarını (gramer) bilmesi olarak adlandırılırken, işlemsel bilgi ise öğrencilerin programlama dilinin kurallarına uyarak kod yazabilmesi olarak ifade edilebilir.

McGill ve Volet (1997), ise programlamada, öğrencilerin kazanması gereken birbiri ile ilişkili üç bilgi tipinden bahseder. Birincisi yazımsal, ikincisi kavramsal üçüncüsü ise stratejik

veya problem çözmedir. Yazımsal bilgi, bir programlama diline ait belirli kuralların bilgisi ve bu kuralları uygulayabilme olarak tanımlanmıştır. Kavramsal bilgi, bilgisayar programcılığının yapıları ve prensipleri olarak tanımlanabilir. Yazımsal ve kavramsal bilgiler gerçek hayat problemlerine çözüm ve tasarım sunabilmeleri için gereklidir. Stratejik bilgi ise genel problem çözme yeteneği olarak tanımlanmıştır.

2.4.5 Programlama öğretiminin önemi. Programlama öğretimi, programlama araçları, programlama dil bilgisi, problem çözme yeteneği, program tasarlamak ve bunu ortaya çıkarmak gibi birçok unsuru içinde barındıran bir süreçtir. Programlama eğitiminde yaygın yaklaşım, öncelikle programlama dilinin temel kavramlarını öğretmek ve ardından öğrencilere programlama işlemi için etkili stratejiler sunmaktır. Üst düzey bilişsel becerilerin kazandırılması için temel kavramların öğretimi oldukça önemlidir (Gültekin, 2006). Öteden beri araştırılan bu konu hakkında 7 yaşındaki gruplarla araştırma yapan Clements ve Gullo (1984), programlama yapan öğrencilerin yansıtıcılık ve farklı düşünce (yaratıcılık) kabiliyeti ile üstbiliş yetenekleri ve yönlendirme yeteneklerinin programlama yapmayanlardan daha yüksek olduğunu tespit etmişlerdir. Diğer taraftan programlamada bir probleme çözüm bulma, analiz yetisi de geliştirilebilmektedir, çünkü problemi alt parçalara ayırmak ve genellenebilir temel bir çözüm oluşturmak gerekmektedir (Saeli, Perrenet, Jochems & Zwaneveld, 2011). Bu bağlamda Ekmel Çetin'in (2012) Ankara'da 17 öğrenci üzerinde yaptığı bir araştırmaya göre bilgisayar programlama eğitiminin uygulanabilir olduğu, programlama eğitiminin çocukların problem çözme becerilerine olumlu yönde katkı sağladığı saptanmıştır.

2.4.6 Programlama öğretiminde yaşanan zorluklar. Yapılan araştırmalarda programlama öğretiminde karşılaşılabilecek sıkıntılar şu şekilde sıralanabilir:

Yerleşmiş alışkanlıklar: Programlama en başından itibaren farklı bilgi ve becerilerin içinde olması gereken bir süreçtir. Sahip olunan önceki akademik alışkanlıklar programlama öğretiminde öğrenci boyutunda geçerliliğini yitirebildiği gibi yeterli gelmeyebilir. Ezberi kuvvetli olan bir öğrenci sadece komutları ezberleyerek problemi çözemez. Okuması çok iyi olan bir öğrenci de yeni bir probleme algoritma geliştiremeyebilir. Kuramsal bilginin uygulama yoluyla bir probleme çözüm bulma becerisi haline gelmesi birçok öğrencide gerçekleşemeyebilir (Ersoy, Madran ve Gülbahar, 2011).

Çoğunlukla yazım kurallarına odaklanma: Lise ve üniversitelerde yer alan bilgisayar programlama dersleri, öğrencilerin bir programlama dilinin komutlarını öğrenmelerini esas almaktadır. Bu durum, öğrenciler için programlama öğrenmeyi zorlaştırmaktadır. Hâlbuki bu dersler problem çözme metotlarını da kapsamalıdır. Problem çözme, onu analiz etmeye ve analiz sonucu farklı çözüm yolları bulmaya bağlıdır. Bu yüzden bilgisayar programlama dersleri, problemleri analiz etmeyi ve algoritmayı kullanarak çözmeyi öğretmelidir (Gilmore, 1990). Programlama dilinin gramer yapısına yoğunlaşma programlama öğretiminde beklenen kazanımdan öğrenciyi uzaklaştırmaktadır.

Seçilen dilin karmaşıklığı: Yaşanan bir diğer zorluk ise programlama öğretiminde bazen algoritma dersleri herhangi bir dil olmaksızın verilirken, bazen de ise bir dil ile birlikte verilmesidir. Eğitim için seçilen programla dili başarıyı etkileyebilir (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko & Miller (1997). Programlama eğitimine başlangıç açısından dil kullanılmaksızın sadece algoritma öğretimi uygun olmasına rağmen, seçilen bir dil ile birlikte algoritma verilecekse, mutlaka, anlaşılması kolay, kuralları basit olan yüksek seviyeli bir dil tercih edilmelidir. Aksi takdirde, algoritma öğrenilebilmekle birlikte, dilin gerektirdiği ekstra kuralların da öğrenilmesi zorluğu ortaya çıkacaktır. Öğrenci kodlamadaki bu zorluklardan dolayı

algoritmaya odaklanamayabilir. Analiz etmek yerine ezberlemek ve hatırlamaya çalışmakla vakit kaybeder. Sonuç olarak öğretim süresince motivasyonu ve başarısı düşebilir Oysaki bu adımda amaç, henüz dil öğrenmek değil, dil öğrenimine temel teşkil eden algoritmayı öğrenmektir. Sonuç olarak seçilen dilin üst bir seviye ve karmaşık bir yapıya sahip olması öğretiminde bir başka önemli zorluk olarak düşünülebilir.

Motivasyon: Programlama öğretiminde derse karşı motivasyon sorunları oluşabilmektedir. Motivasyonunu yitirmiş bu öğrencilerin öğrenmeyi gerçekleştiremeyeceği ve en nihayet başarısız olabileceği belirtilmiştir (Jenkins & Davy, 2002). Programlama öğretimine yardımcı araçlarda yer alan görsel özelliklerin önemini vurgulamak amacıyla yapılan araştırmaların sonuçları, görsel özelliklerin öğrenci başarısı ve motivasyonunda önemli bir etkisi olduğunu göstermektedir (Kelleher, Pausch & Kiesler, 2007).

Yeni bir konu olması: Programlamaya giriş dersi genellikle, lise düzeyinde ilgili meslek liselerinin Lise 1 sınıfında veya üniversitelerde ilgili bölümün 3. veya 4. döneminde okutulmaktadır. Söz konusu dönemlerde gerek genel hatlarıyla bilgisayar konusunun gerekse programlama konusunun çoğu öğrenci için yeni olması ve ders kapsamındaki konuların ağırlıklı olarak soyut kavramlar üzerine inşa edilmesi sebebiyle, ders hem öğreten hem öğrenen için oldukça kompleks bir yapıya sahiptir ve bu nedenle genel olarak öğrencilerin başarı ortalamaları pek yüksek olmamaktadır (Erdoğan, 2005). Proulx'in (2000) çalışması bu görüşü destekler niteliktedir. Proulx'a göre; yeni başlayan öğrenciler için bilgisayar programlama ile ilgili kavram ve bilgileri anlamak zor olduğunu vurgulamaktadır.

Buna karşın Casey (1997), Calder (2010), Kaucic ve Asic'in (2011) çalışmalarına bakıldığında erken yaşlarda programlama eğitimi alan bireylerin üst düzey programlama dillerine

geçtiklerinde daha kolay adapte oldukları ve program yazmada daha başarılı oldukları belirlenmiştir.

Soyut kavramlar içermesi: Programlama dili öğretilirken çoğu işlem ve kavram öğrencilere soyut gelmekte öğrenciler bu bilgileri somutlaştırmakta zorlanmaktadırlar (Ersoy, Madran ve Gülbahar, 2011). Programlamanın soyut doğası çoğu öğrencinin program yapısının nasıl çalıştığını ve problemlerin nasıl çözüme kavuştuğunu görmelerini ve anlamalarını zorlaştırabilmektedir (Esteves & Mendes, 2004). Örneğin, Meisalo ve arkadaşları çalışmalarında uzaktan eğitim öğrencilerinin %32'sinin henüz eğitimin birinci yılında bilgisayar dersini devamsızlık yoluyla bıraktığını tespit etmiştir ve buna sebep olarak öğrencilerin dersteki uygulamalar veya teorik yönlerin çok zor bulmasını göstermiştir (Meisalo, Suhonen, Torvinen & Sutinen, 2002).

Yaklaşım tarzı: Belki de önemli hususlardan birisi de ezberci mi yoksa sistematik bir yaklaşımla mı öğretimin yapılması gerektiğidir. Farklı kalıpların dışına çıkılmayacaksa, benzer programlama dilleri için ezberci bir yaklaşım çözüm olabilir. Diğer taraftan farklı problemlere farklı çözümlerin olabileceği durumlarda sistematik yaklaşım esas alınabilir.

Öğretim teknikleri: Klasik öğretim teknikleri de programlama başarısını etkileyebilir (Jenkins, 2002). Araştırmalara göre programlama öğrenmeye yeni başlayan öğrenciler, genellikle programlama mantığını kavramakta güçlük çekmektedir. Programlama öğretiminde geleneksel öğretim metotları, öğrencilerin programlama öğrenme konusundaki sıkıntılarını gidermede başarılı olamamaktadır. Programlama becerileri en iyi deneyimle kazandırılabilmektedir (Traynor & Gibson, 2004). Programlama öğrenimi diğer derslere nazaran daha pratik yaklaşımları olan ve sadece teorik değil birçok pratik bilgiye de bağımlı olan bir konudur (Esteves & Mendes, 2004).

Bu araştırmada ilerleyen başlıklarda programlama öğretiminde kullanılabilecek yöntemlerden bahsedilecektir.

Yukarıda bahsedilen maddelerde yaşanan zorluklara bakıldığında programlama öğretiminin eğitim yazılımları ile (görselleştirme araçları) öğretilebileceği gibi bir alternatif yola eğitimcileri sevketmiştir. Öğrencilerin programlama öğrenmelerini kolaylaştırmak amacıyla çeşitli görselleştirme araçları geliştirilmiştir. Bu araçlar, programlama öğrenmeye yeni başlayanların etkileşimli bir ortamda basit komutlar kullanarak temel düzeyde programlar oluşturmalarını sağlar. Yapılan araştırma sonuçlarına göre programlama öğretiminde görsel araçların kullanımı, öğrenci başarısını artırmaktadır (Hyrskykari, 1993; Malan & Leitner, 2007; Peppler & Kafai, 2007; Hu, 2004; Cooper et al., 2006).

2.4.7 Programlamayla ilgili yanılgılar. Programlama alanında uzmanlaşmış insanlar arasında, her zeki insanın programcı olabileceğine dair bir kanı vardır. Fakat yapılan araştırmalar ve sınıf deneyimleri, bu varsayımın doğru olmadığını göstermektedir. Nitekim programlama becerisi tek başına salt ve somut bir IQ düzeyine değil, bilgisayar tutumu, genel yetenek, genel akademik başarı, matematik başarısı, programlama kavramıyla ilişkili ilgi ve yetenekler, soyut düşünebilme becerisi, detaylara odaklanabilme ve konsantrasyon düzeyi, öğrenim görülen okulun türü ve cinsiyet gibi pek çok farklı faktöre bağlıdır. Bu konuda yapılan araştırmalar incelendiğinde programlama konusunda başarılı insanların yukarıda bahsedilen özelliklerin birçoğuna sahip olduğunu göstermiştir. Bir başka husus ise, yine çeşitli araştırmalara bakıldığında diğer alanlarda çok iyi olan öğrencilerin programlamada başarısız olabildiği de görülmektedir. Dolayısıyla programlama becerisi diğer bilgisayar becerilerinden bağımsız, ayrı bir beceri olarak algılanmalıdır (Erdoğan, 2005).

Perry (2009), bilgisayar programlamayla ilgili olarak yaygın olan üç adet yanılgıdan bahsetmektedir. Birinci olarak, sadece matematik becerisi iyi olan insanların programcı olabileceği, iyi programlar yazabileceğidir. Tam tersine bilgisayar, matematiği insanlar için yerine getirir. Üstelik programlama alanında gelişmek matematik zekâsını da geliştirmektedir. İlgili literatüre bakıldığında programlama becerisinin matematik problem çözme becerisinde olumlu artış sağladığı görülebilir (Calder, 2010). İkinci olarak, bilgisayarın programları yanlış yapabilme ihtimalidir. Bilgisayar, sadece girilen komutları işlemeye yönelik çalıştığı için ortada bir hata varsa, bunun programın yazımından kaynaklanıyor olma ihtimali yüksektir. Son olarak da bilgisayar programlamanın zor olduğu yanılgısı bulunmaktadır. Hâlbuki bilgisayara ve teknik konulara ulaşmanın sıradanlaştığı 2000'li yıllarda programlama dillerini öğrenmek ve program yazmak da kolaylaşmıştır. Önceleri siyah ekranda satır satır komutlar yazılırken artık bu iş yeni nesil programlama dilleriyle daha kolay bir hale gelmiştir. Bu çalışmanın da içeriğinde bir sonraki bölümde bahsedilecek olan çocuklar için hazırlanan eğitim yazılımları, konuyla ilgili zorlukları en alt seviyeye indirmiştir.

2.4.8 Türkiye'de ve dünyada programlama öğretimi. Gerek mesleki yeterlilik için olması gerekse bilgisayar okuryazarlığına katkısı bakımından programlama dersleri yaygınlaşmaktadır. Araştırmalar incelendiğinde programlama öğretimine kimi ülkelerde lise kademelerinde kimi ülkelerde ise ilköğretim kademelerinde başlandığı gözüküyor.

Programlamanın temellerinin küçük yaşlarda öğretilmesinin gerekliliğine karşın, bazı ülkelerde programlama ile ilgili konuların ilköğretim öğretim programına henüz eklenmemiş olması, önemli bir eksiklik olarak karşımıza çıkmaktadır. İsrail ve Kanada'da ise programlama eğitimi ile ilgili dersler liselerde verilmektedir (Tucker et al., 2003). İsrail'de programlama öğretimi, 10. 11. ve 12. sınıflarda bilgisayar bilimleri dersi kapsamında verilmeye

başlanmaktadır. Bu ders öğretim programı kapsamında algoritmalar ile ilgili temel düzeyde bilgi verilmektedir. Kanada'da ise programlama ile ilgili bilgiler, ortaöğretim kurumlarında bilgisayar mühendisliği ve bilgisayar bilimleri ile ilgili açılan dersler kapsamında verilmektedir (Stephenson, 2001).

Öte yandan birçok ülkede, öğrencilere programlama ile ilgili temel bilgiler ilköğretim okullarında verilmektedir. Tayvan'da son yıllarda ilköğretim okullarında Lego Mindstorms'a yer verilmeye başlanmış (Lin, Yen, Yang & Chen, 2005), Kore'de ise 2010 yılında ortaokulların, 2011 yılında ise liselerin bilişim öğretim programında değişiklikler yapılması kararı alınmıştır (Choi, Bell, Jun & Lee, 2008).

Hindistan, bilindiği gibi programlama ile ilgili büyük bir pazar payına sahiptir. Ülke, birçok başarılı bilgisayar mühendisi yetiştirmektedir. Bu başarının nedenlerinden biri olarak ilköğretim çağından itibaren öğrencilere programlama ile ilgili bilgi ve becerilerin verilmesi düşünülebilir. Hindistan eğitim sisteminde K-12 düzeyinde yer alan bilgisayar bilimleri öğretim programında programlama konuları aşağıdaki biçimde yer almaktadır:

- 3.sınıfta öğrencilerin temel programlama komutları ile ilgili bilgi edinimi amaçlanmaktadır.
- 4.sınıfta öğrencilerin LOGO programlama dilinde sık kullanılan komutlar kullanılarak işlemler yapmaları amaçlanmaktadır.
- 5.sınıfta öğrencilerin LOGO programlama dili ile programlamaya başlangıç yapmaları amaçlanmaktadır.
- 6.sınıfta öğrencilerin BASIC programlama dili kullanarak programlama yapmaları amaçlanmaktadır.

- 7.sınıfta öğrencilere yapısal programlama, sabitler, değişken ve döngü kavramları ile ilgili bilgiler verilmektedir. Öğrencilerin BASIC programlama dili kullanılarak uygulamalar yaptırılmaktadır.
- 8.sınıfta ise yapısal programlama ile ilgili kavramlar genişletilmektedir. BASIC dili kullanılarak daha önce öğrenilen sabit, değişken, döngü gibi konularda alıştırmalar yaptırılmaktadır.
- 9. ve 10. sınıfta ise öğrencilere fonksiyon, sınıf ve dizi kavramları ile ilgili bilgiler verilmektedir (Mandir, 2007).

Tucker ve diğerleri (2003), araştırmalarında Avrupa, Asya, Güney Afrika, Rusya, Avustralya ve Yeni Zelanda'da bilgisayar bilimleri dersinin K-12 öğretim programına eklendiğini belirtmektedirler. K-12 düzeyinde Bilgisayar Bilimi dersleri için geliştirilen ACM (Association for Computing Machinery) öğretim programına göre özellikle 8. sınıf öğrencileri bir görevi tamamlamak için kullanılabilecek uygun adımlarla algoritmik problem çözmeyi öğrenmeleri beklenmektedir. Bu adımlarda koşul ifadeleri ve tekrarlar kullanılabilir (Lin, Yen, Yang & Chen, 2005).

Ülkemizde, programlama ile ilgili temel kavramlar ilköğretim okullarında verilmektedir. İlköğretim bilişim teknolojileri kitabının 6–8. Basamaklarında ele alınan kazanımlar arasında "Basit bir program için algoritma ve akış şeması oluşturma" ve "nesne tabanlı programlama dilinde nesneleri kullanarak basit programlar oluşturma" hedef ifadeleri yer almaktadır (İnce, Şenyüzlü ve Uğur, 2007). Buna göre bilişim teknolojileri dersinin amaçları arasında öğrencilere algoritma geliştirme becerilerini kazandırabilmek yer almıştır. Bu yolla öğrencilere problemlere doğru yaklaşımlar sergileyerek uygun çözüm yolu geliştirme becerilerinin kazandırılması amaçlanmaktadır.

2.4.9 Programlama öğretimi yazılımlarının sınıflandırılması. Raeder'ın (1985) yaptığı bir araştırmada, insan zihninin metin tabanlı kaynakları okumak yerine, görsel bilgileri daha iyi anlamlandırıp bunlar arasındaki ilişkiyi daha sağlam kurduğunu tespit etmiştir.

Kullanılacak olan görselleştirme şekli konuya bağlı olarak resim veya animasyon olabilir. Animasyon olarak geliştirilen görselleştirme aracı, sunulacak verinin oldukça büyük veya karmaşık olması, hareketlendirmesi gerekliliği ve kavramların birbirleriyle olan ilişkilerini açıklama durumlarında kullanılmalıdır. Görselleştirme aracı sınıf sunumları, laboratuvar uygulamaları ve ödevlerde kullanılabilir. Görselleştirme aracının asıl gücü soyut kavramları şekillerle desteklemesidir. Konu başında, sonunda veya konu içerisinde tartışma ortamı yaratmak için kullanılabilir. Öğrencilere farklı bakış açıları kazandırmaya ve çok boyutlu soyut kavramları anlamlandırabilmeyi sağlar (Bergin & Martinez, 1996).

Bergin ve Martinez'e (1996) göre programlama öğretiminde üç farklı görselleştirme türü vardır. Bunlar:

1. **Program Görselleştirme:** Bu tip görselleştirme aracı, bir programın ve onun verilerinin nasıl çalıştığını grafiksel olarak sunmak için kullanılır. Veri ve kodların önceden tanımlı görselleştirmeleri vardır. Kullanıcı ile etkileşim, programın istenilen yerde durdurulması ve girdilerde değişiklik yapılabilmesi şeklinde sağlanmaktadır. Rajala, Laakso, Kaila ve Salakoski (2008), araştırmalarında program görselleştirme araçlarının, öğrencilerin programların davranışlarını anlamalarına yardımcı olduğunu ifade etmektedirler. Araştırmalarında program görselleştirme araçlarına örnek olarak Ville, Jeliot3 ve BlueJ araçlarını sunmuşlardır.
2. **Algoritma Animasyonları**: Temel algoritmaların hangi işlemleri tamamladığını görseller yardımıyla göstermek amacıyla kullanılır. Algoritma animasyonları, algoritmanın bir görsele

yönlendirilmesi ve algoritma yürütüldüğünde işlem sırasına göre bu görselin hareket ettirilmesi mantığı üzerine çalışır. Görselleştirme, kod ve verilerle sınırlı değildir. Algoritma animasyonlarında kullanıcı, neyin görselleştirileceğine kendisi karar verir. Kullanıcı, açıklayıcı notlar yardımıyla algoritma ve nesneler arasındaki ilişkiyi tanımlayabilir ve ilgili görselleri seçerek ne zaman görselleştirileceğini ayarlar. Algoritma animasyonlarında kullanıcı etkileşimi, programcı tarafından tanımlanabilmektedir. Bu tür araçlar, kullanıcı ile yüksek düzeyde etkileşim sağlamaktadır. Bu araçlara örnek olarak Xtango, Balsa, Samba, The Sort Animator, JHAVE verilebilir.

3. **Veri Görselleştirme:** Bu tip görselleştirme araçları ise program kodu içerisinde veri tiplerinin neler olduğu ve hangi işlemlerden sonra son durumlarının ne olduğu hakkında bilgi verir. Veri görselleştirme araçları, veri yapıları üzerinde çok genel işlemler olan veri ekleme çıkarma gibi işlemler için kullanılmaktadır. Bu tür yazılımlarda kullanıcı ile düşük düzeyde etkileşim sağlanmaktadır. Chen ve Sobh (2001), araştırmalarında JavaMy adında bir veri görselleştirme aracını tanıtmışlardır.

Cooper ve diğerleri de (2006), araştırmalarında programlama öğretiminde kullanılan görselleştirme araçlarını kategorilere ayırmışlardır:

1. **Algoritmayı hikâyeleştiren araçlar:** Programlamayı bir animasyon karakter yardımıyla hikayeleştirerek öğretmeyi amaçlayan araçlardır. Storytelling (hikayeleştirme) kullanımının iki önemli faydası vardır: Birincisi öğrencilerin kendi filmini yönetebilmeleri, onların ilgisini çekmektedir. İkincisi ise öğrencilerin programı hikâye şeklinde oluşturmaları, onların görsel senaryo ile tanışmalarını sağlamaktadır. Alice ve Jeroo, bu tür araçlara örnek olarak gösterilebilir. Bu tür araçlardan JPie, hareketli bir karakter aracılığıyla sürükle-bırak yöntemini kullanarak öğrencilere programlama öğretmeyi amaçlar (Cooper et al., 2006). Bu araştırmada

kullanılan Microsoft Kodu Game Lab aracı da bu sınıfa dâhil edilebilir. Bu tür yazılımların kullanımının programlama öğretiminde sağladığı kolaylıklar aşağıda sıralanmaktadır.

- Kod yazımı olmadığı için yazım hatalarından kaynaklanan problemler azalır.
- Veri ve nesne kavramları, görselleştirme ile öğretilir.
- Program değişiklikleri görülebilir. Basit biçimde hata ayıklama işlemi yapılabilir.
- Programlama öğretimi, önce senaryo (storyboard) sonra programlama mantığı ile verilebilir (Klassen, 2006).

2. **Görsel programlama araçları:** Programları bir ara yüz yardımıyla, sürükle–bırak biçiminde yapılandırmayı sağlayan araçlardır. Görsel programlama araçları, programcıkların yazım hataları ile ilgili sorunlar olmaksızın geliştirilebilmesine imkân vermektedir. Bunun yanı sıra kullanıcıların, programlama içeriklerinin grafiksel gösterimleri üzerinde değişiklikler yapabilmesini sağlamaktadırlar. Görsel programlama araçları, ayrıca programların çalıştırılıp dönüt alınabilmesini sağlamaktadırlar. Bu araçlara örnek olarak JPie, Alice ve Karel Universe gösterilebilir (Cooper et al., 2006). Bu araştırmada kullanılan Scratch aracı da bu başlık altında düşünülebilir.
3. **Akış şeması modelli araçlar:** Program parçalarını işlem sırasını gösterecek biçimde birbirine bağlayarak kodları yapılandırmayı sağlayan araçlardır. Bu araçlar, yazım hatası karmaşasını azaltmaları ve öğrencilerin eksik satırlar yerine problem çözmeye odaklanmalarını sağlamaları bakımından kullanışlıdırlar. Raptor, FLINT, Iconic Programmer ve Visual Logic bu tür araçlara örnek olarak gösterilebilir (Cooper et al., 2006). Crews ve Ziegler'in (1998) akış şemasının kullanımının programlama öğrenmeye yeni başlayan öğrenciler üzerindeki etkisini araştıran çalışmalarında, öğrencilerin akış şeması kullandıklarında daha az hata yaptıkları,

özellikle basit problemlerin çözümünde daha başarılı olduklarını ve kendine daha fazla güvendikleri sonucu ortaya çıkmıştır.

4. **Program çıktılarını görselleştiren araçlar:** Oluşturulan programlar yürütüldüğünde çoklu ortam kullanarak dönüt veren araçlardır. Bu araçları kullanarak, öğrenciler kendi programlarını kodlayıp test edebilmektedirler. Daha da önemlisi öğrenciler, anında görsel dönüt alabilmektedirler. Lego Mindstorms ve JES bu tür araçlara örnek olarak gösterilebilir (Cooper et al., 2006). Bu araştırmada kullanılan Robomind aracı bu sınıfta düşünülebilir.
5. **Sıraya dizilmiş programlama dili araçları:** Programlama dilini öğrenci seviyesine göre uygun biçimde aşamalı olarak öğreten araçlardır. Programlama öğretimi, yazım kuralı ve karmaşıklığın en aza indirgenmiş şekli ile başlar. Programda yer alan hata mesajları, öğrenciler ilerledikçe seviyelerine uygun biçimde değiştirilir. Daha sonraki aşamada programlama diline yeni ve daha karmaşık özellikler eklenir. Başlangıç düzeyinde öğrenciler, sadece belirgin özelliklere odaklanırken, programlama diline yeni özellikler eklendikçe aşamalı olarak karmaşık özellikler de öğrenmeye başlar. Bu tür araçlara ProfessorJ ve RoboLab örnek olarak gösterilebilir (Cooper et al., 2006).

Pek çok programlama dili ve programlama mantığının öğretilmesinde karşılaşılan güçlükler göz önüne alındığında uygulanmış farklı yaklaşımlar vardır. Örnek olarak hali hazırda kullanılan dilin "zor" olmasına karşılık bunun yerine daha basit diller (mini language) önerilmiştir. Bu tür diller yazılım geliştirmekten ziyade programlama dili öğretimi amacıyla geliştirilmiş, basit komutlardan ve farklı etkileşim tekniklerine sahip dillerdir (Ersoy, Madran ve Gülbahar, 2011). Bu araştırmada kullanılan Microsoft Small Basic aracı da basit diller sınıfına örnek olarak verilebilir.

2.4.10 Programlama öğretiminde kullanılabilecek yöntemler. Korhonen (2003), programlama dersinde öğrenme ortamının nasıl düzenlemesi gerektiği ile ilgili yaptığı çalışmasında, öğrencilerin görselleştirme araçları kullanarak etkileşimli biçimde kendilerine verilen program örneklerini düzenleyebilmeleri ve kendi tasarımlarını yapabilmelerinin önemli olduğunu vurgulamaktadır. Buna göre, programlama öğretimi özetle beş farklı yöntemle yapılabilmektedir:

1. Öğretici materyallerin kullanımı;
2. Görselleştirme aracının kullanımı;
3. Sınıf ortamında öğretmen rehberliğinde laboratuvar uygulamalarının yapılması;
4. Öğrencilerin Internet ortamında kendi hızlarına göre ilerleyebilmeleri;
5. Yukarıdaki maddelerin hepsinin beraberce kullanımı.

Yukarıdaki maddeler bir önceki maddeyi takiben de kullanılabilir.

Naps ve diğerlerine (2003) göre bilgisayar bilimleri dersinde kullanılan görselleştirmeler, öğrencileri aktif olarak öğrenmeye dâhil etmelidir. Aksi takdirde katmıyorsa nasıl tasarlanmış olursa olsun eğitimsel değeri düşük olacaktır.

Yapılan bir araştırmaya (Hu, 2004) göre programlama öğretiminde önce konuların genel hatlarıyla verilmesi daha sonra uygulamaların yapılması yerine teori ve pratik uygulamanın iç içe kullanımı daha başarılı olmaktadır. Bu bağlamda öğretime programlama dillerinden bağımsız biçimde başlanmalı, daha sonra bir programlama dili kullanılarak uygulama yapılmalıdır. Programlama mantığı öğretilirken, basitten karmaşığa doğru adım adım gidilmesi gerektiği unutulmamalıdır.

Naps ve diğerleri (2003) programlama öğretiminde görsel teknolojinin kullanımı ile ilgili yaptıkları taksonomiye göre programlama öğretimi altı adımda gerçekleşmektedir:

1. Görselleştirme aracı olmaksızın anlatım;
2. Görselleştirme aracının kullanımı;
3. Yanıtlama;
4. Değişiklik yapabilme;
5. Kendi programını oluşturma;
6. Sunma.

Araştırmacılar, çalışmalarının sonucunda programlama öğretiminde etkileşimin önemini vurgulamaktadırlar.

Garner (2003), yazılım döngüsünün 4 temel aşamasını; a) Problemin analizi, b) Çözüm veya algoritma dizayn etmek ve geliştirmek, c) Algoritmayı uygulamak, d) Algoritmayı test veya revize etmek, kullanarak görselleştirme araçlarının hangi aşamalarda kullanılabileceği ile ilgili örnekler vermiştir. Buna göre Garner, akış şeması yazılımlarının program tasarım, geliştirme ve algoritma oluşturma aşamalarında; algoritma animasyonlarının ise programın test etme ve gözden geçirme aşamasında kullanılabileceğini ifade etmiştir.

Bergin ve Martinez'e (1996) göre görselleştirme araçları, eğiticilere yeni eğitim yöntemleri olanağı sunup, öğrencilerin ilgilerini çektiğinden bu alanın eğitmenlerine örnek yaratma açısından oldukça yardımcı olmaktadır. Bu da hem eğitmenlerin hem de öğrencilerin motivasyonlarını daha da arttırıcı bir etkiye sebep oluyor. Lin ve Zhang (2003), çalışmalarında programlama öğretiminde kullanılan yöntem ve tekniklerin öğrenci motivasyonu üzerindeki etkisini araştırmışlardır. Öğrenci merkezli yaklaşım, işbirliğiyle öğrenme, algoritma tasarlama, yardımcı bir yazılım kullanımı, animasyon oluşturma ve erişilebilir kodlardan yararlanma bu

yöntem ve tekniklerdendir. Çalışma sonucunda, öğrencilerin kendi animasyonlarını oluşturmalarının onların motivasyonunu artırdığı sonucuna ulaşılmıştır.

Motivasyon açısından aşağıdaki önermeler de göz önünde tutulmalıdır (Bergin & Martinez, 1996):

- Karmaşık kavramların sadeleştirilmesi ve anlaşılabilmesi için resimler ve görsellerden yararlanılmalıdır.
- Ayrıca görselleştirme araçları eğitmenlere daha az zamanda daha etkili ve içerikli materyaller geliştirmek için de yardımcı olur.
- İyi tasarlanmış benzetimler öğrencilerin anlama düzeylerini artırır.

Hundhausen, Douglas ve Stasko (2002) çalışmalarında programlama öğretiminde görselleştirme araçlarının etkileşimli kullanımı için çeşitli senaryolara yer vermişlerdir. Bu senaryolar şunlardır:

1. Algoritmanın nasıl çalıştığının anlaşılması için grafiksel gösterime başvurulabilir.
2. Öğrenciler, kendi görselleştirmelerini oluşturabilirler.
3. Öğrenciler, kendi hazırladıkları görselleştirmeleri arkadaş ve öğretmenlerine sunarak onlardan fikir alabilirler.
4. Öğrenciler laboratuvar alıştırmalarını yaparken etkileşimli bir biçimde algoritmaları keşfedebilir.
5. Öğrenciler, kendi görselleştirmelerini çizerek, başkalarının hazırladıkları görselleştirmeleri kâğıt üzerinde inceleyerek ya da etkileşimli bir görselleştirme yazılımı kullanarak çalışabilirler.
6. Öğretmenler, öğrencilerin hatalarını tespit edebilir ve sorularına cevap verebilirler.

7. Öğrencilerin bilgileri değerlendirilirken, öğrencilerin algoritmaları isimlendirmeleri, algoritma içerisindeki görsellerin tanımlamaları ve algoritmanın nasıl çalışacağını tahmin etmeleri ya da çizmeleri istenebilir.

Nelson ve Rice (2000) ise, yaptıkları araştırmada programlama öğretiminde öğrencilerin programlama dilinden bağımsız biçimde algoritma oluşturabilmelerinin önemine değinmişlerdir. Çalışmada ayrıca öğrencilerin yazdıkları kodlarla ilgili dönüt alabilmelerinin önemli olduğu vurgulanmıştır.

2.4.11 Programlama öğretimi yazılımları. Günümüzde programlama öğretiminde gerek bilgisayar teknolojisinin ilerlemesi gerekse de internetin yaygınlaşmasıyla birçok eğitim materyali geliştirilmiştir. Bu materyallerde algoritmaları görselleştiren, çeşitli animasyon teknikleri ile programlama mantığını doğru bir zemine oturtmaya çalışan öğeler mevcuttur. Bunlar için şekil, ses, video gibi çoklu ortam unsuları bulunabilir. Bu tip materyallerin asıl amacı alışıla gelmiş öğretimde yaşanan zorlukları bu öğelerle aşabilmektir. Aşkar'a (1990) göre, emek verilerek hazırlanmış iyi bir program gereksinimlere cevap veren bir programdır. Bazı öğrenciler çok hızlı öğrenip öğretmenin vermeyi hedeflediğinin çok ilerisine gidebilirler. Öte yandan bazı öğrenciler ise yavaş öğrenirken ve öğrenme gereksinimleri diğerlerine göre daha farklı olabilir. Hazırlanan öğretim aracı bu tür ihtiyaçları karşılayabilirse uygulamaların başarılı olma olasılığı artar.

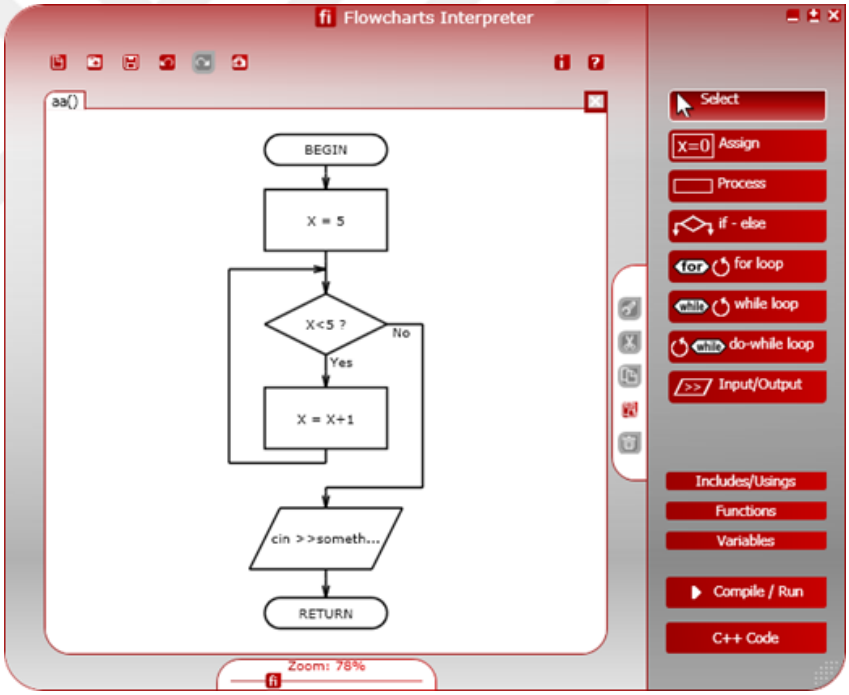
Literatürde yer alan programlama öğretim araçları sınıflandırması çerçevesinde incelenen çeşitli programlama öğretim yazılımlarını şu şekilde sıralayabiliriz:

FLINT

Akış diyagramları yardımıyla görsel algoritmalar tasarlamak için geliştirilmiş bir araç olan FLINT (FLowchart INTerpreter) programlama dilinin yazımsal kurallarının detaylarından uzaklaşarak öğrencilerin görsel arayüzler yardımıyla akış diyagramları geliştirmelerini sağlayan bir araçtır. Yapılan algoritma tasarımı çalıştırılırken aynı zamanda istenirse C++ diline göre çıktı alınabilmektedir.

Şekil 2.5

FLINT Yazılımı Ekran Görüntüsü

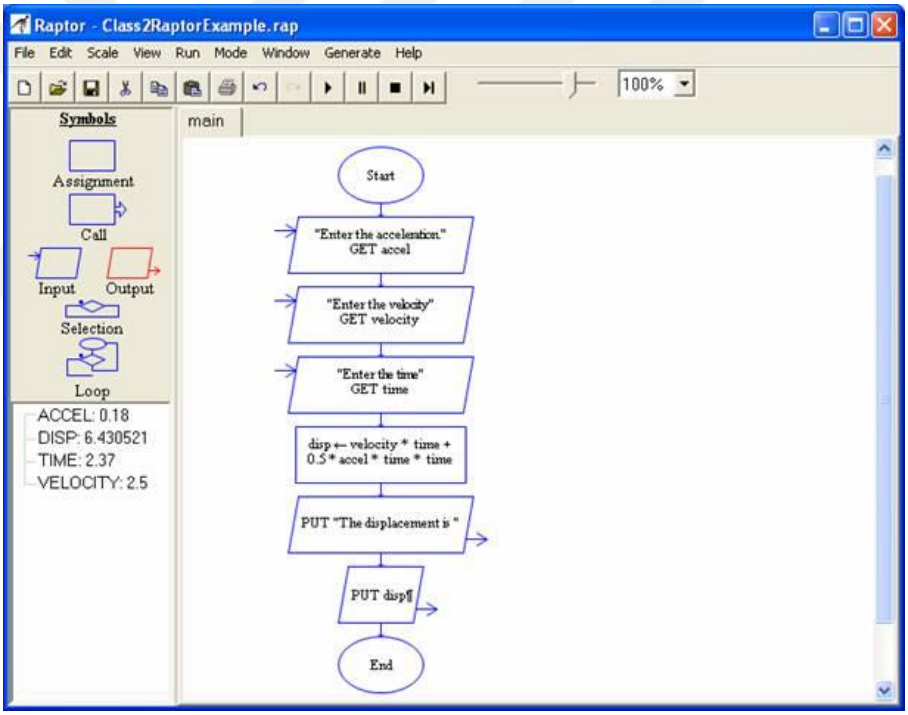


Raptor

Raptor yazılımı ise Birleşik Devletler Hava Kuvvetleri Akademisi (U.S.A.F.A) tarafından geliştirilmiştir. Bu yazılım akış diyagramları oluşturmada yardımcı olmaktadır. Sol bölümde bulunan algoritma parçalarını çalışma alanına taşıdıktan sonra ilgili parçaların içlerine gereken komutların yazılmasıyla bir akış çizelgesi meydana getirilir.

Şekil 2.6

Raptor Yazılımı Ekran Görüntüsü



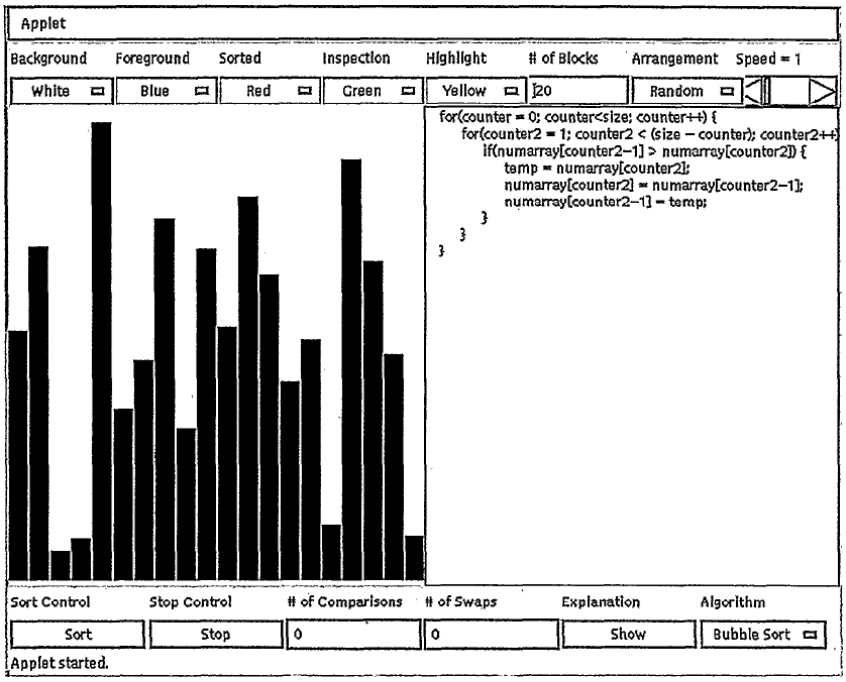
The Sort Animator

Bir "algoritma görselleştirme" aracı olan bu araç Dershem ve Brummund (1998) tarafından geliştirilmiştir. Java Platformunda Applet teknolojisi kullanan bu araçta hem algoritma kodunu hem de bu kodun animasyonunu aynı anda görebilmek mümkündür. İşlem sırasına göre

eşzamanlı olarak sıra hangi komutta ise o komut renklendirilip bu kodun sonucu olan animasyon da görüntülenir.

Şekil 2.7

The Sort Animator Yazılımı Ekran Görüntüsü

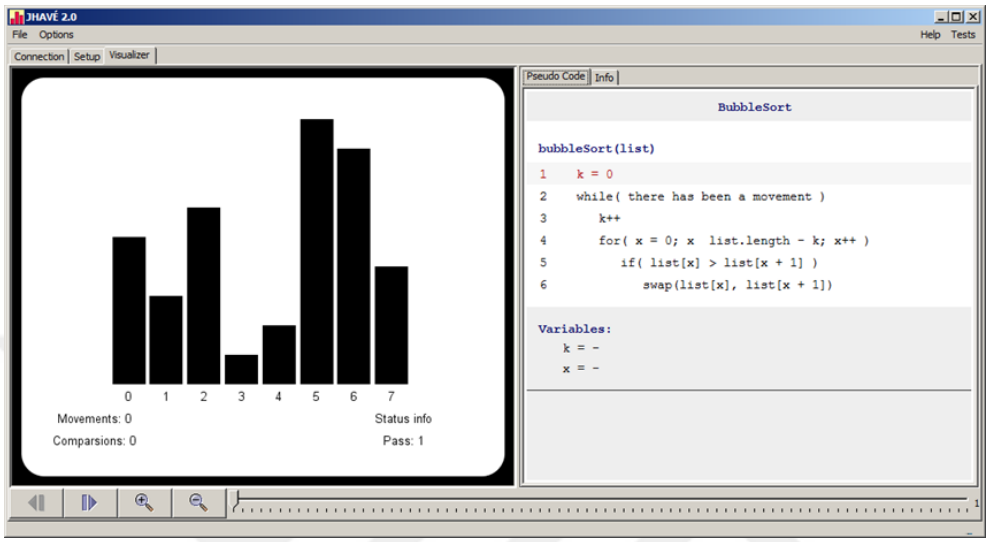


JHAVE

Naps, Eagan ve Norton (2000) tarafından geliştirilen JHAVE (Java-Hosted Algorithm Visualization Environment) isimli istemci-sunucu mimarisine dayanan "algoritma animasyonları" aracı indirilebildiği gibi *http://csf11.acs.uwosh.edu* web adresi üzerinden de çalıştırılabilmektedir.

Şekil 2.8

JHAVE Yazılımı Ekran Görüntüsü



Araştırmaya göre JHAVE programının faydaları şunlardır:

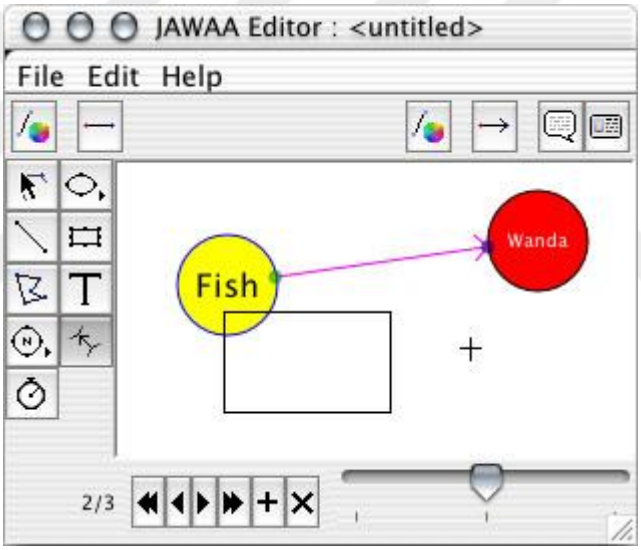
1. Öğrencilerin adım adım ilerleyerek çalışmalarını sağlar.
2. Bilgi ve kod pencereleri sayesinde öğrencilerin algoritmayı oluşturan kodları anlamalarına da yardımcı olur.
3. Nesneler yardımıyla öğrencilerin öğrenmesini kolaylaştırır.
4. JHAVE yardımıyla öğrencilere bir süre sonra hangi ekranın geleceği ile ilgili sorular sorularak öğrencilerden tahmin yürütmeleri istenebilir.
5. Tasarımcılar, hem grafiksel görüntüleme hem de etkileşimli araçlar oluşturabilir (Naps, (2005).

JAWAA

Bir başka algoritma görselleştirme aracı ise Duke Üniversitesi'nde bilgisayar bilimleri derslerinde öğrencilerin görsel olarak kavramları öğrenebilmesi için kullanılan JAWAA'dır. Yazılan programın algoritma animasyonunu çıkaran bu araç yazılan komutları HTML diline göre animasyonlara çevirmekte ve internet üzerinden çalıştırılabilmektedir (Rodger, 2002).

Şekil 2.9

JAWAA Yazılımı Ekran Görüntüsü



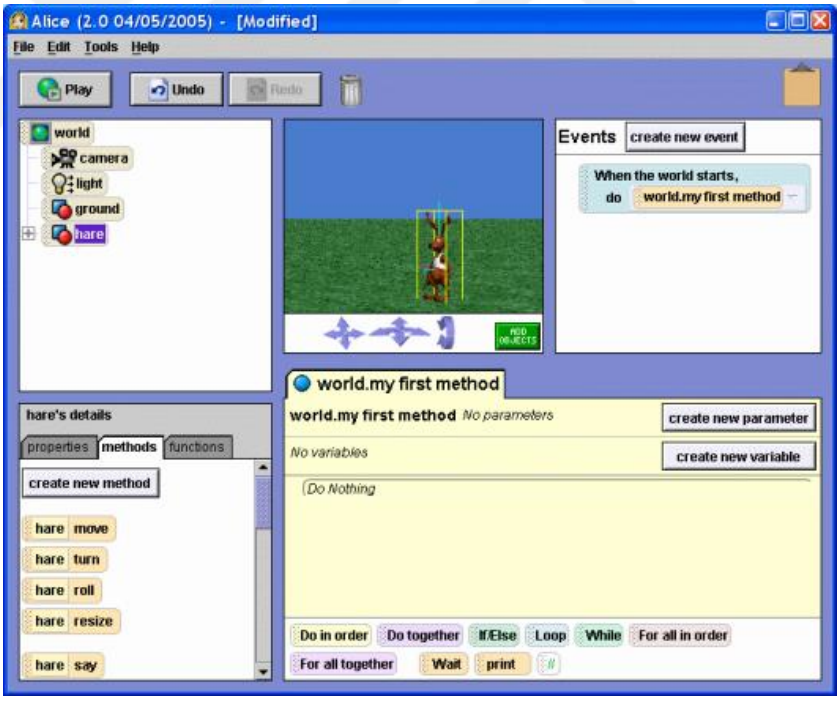
Alice

Algoritma animasyonlardan başka programlama mantığını görsel bir şekilde aktarabilmek içinde belirli bir kahramanın olduğu ve değişik başka özel araçlarla desteklenmiş öğretim yazılımları da mevcuttur. Alice, bir öğrencinin nesne tabanlı programlama ile ilk bilgilerini elde edebilecek şekilde tasarlanmış bedava kullanılabilir bir öğretim aracıdır. Öğrencilerin animasyon film ve basit video oyunları yaratma bağlamında temel programlama kavramlarını öğrenmesini sağlar. İnteraktif arayüzünde kullanılan standart ifadeler üretime dayalı Java, C++ ve C# gibi programlama dillerine benzer bir yapıda olan Alice yazılımı Carnegie Mellon Üniversitesinde

görev yapan Randy Pausch tarafından geliştirilmiştir. Alice'in geliştirilme amacı bilgisayar programlamasına yeni başlayanlar için ilgi çekici üç boyutlu grafik animasyonlar geliştirebilmeyi kolaylaştırmaktır (Dann, Cooper & Pausch, 2000). Alice, hazırlanan programın hemen çalıştırılmasına izin vererek öğrencilerin kendi hazırlamış oldukları animasyonlardaki nesnelerin davranışları ile programlama ifadeleri arasındaki ilişkiyi anlamalarını sağlar. Kendi sanal dünyalarındaki nesneleri değiştirerek öğrencilerin programlamaya giriş dersinde öğretilen yapılar üzerinde deneyim kazanmasını sağlar (Alice3D, 2015).

Şekil 2.10

Alice Yazılımı Ekran Görüntüsü

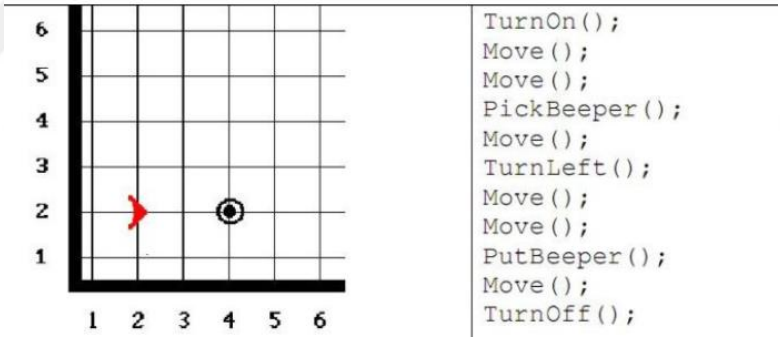


Karel the Robot

Kodlamayı basit kelime ve cümlelerle öğretmeye çalışan “Karel the Robot” yazılımı Richard E. Patis tarafından geliştirilmiştir. Programda sıkıcı ve öğrenmesi zor olan programlama terimleri yerine basit ve eğlenceli anlaşılır komutlar tercih edilmiştir. 1980’den beri çeşitli formatları yapılmış olan ve robotlar tarafından yapılması sağlanan çeşitli problem setlerinin tanımlandığı bu yazılımda öğrenciler tarafından yeni problemlerin tanımlanmasına da olanak sağlanır. Bu sayede öğrenenlere basit ve eğlenceli bir öğrenme ortamı sunulmuştur.

Şekil 2.11

Karel the Robot Yazılımı Ekran Görüntüsü



StageCast Creator

Stagecast Creator, yeni başlayan programcılar için görsel arayüze sahip bir sistemdir. Bu programın amacı öğretmen ve öğrenciler için programlama simülasyonlarını oluşturup bunları değiştirmelerini sağlamaktır. Stagecast Creator programlama işlemlerini direk olarak değiştirmeye yarayan kuralları içerir ve benzeşime dayalı sunumlar için kullanılır (Smith, Cypher & Tesler, 2000).

Şekil 2.12

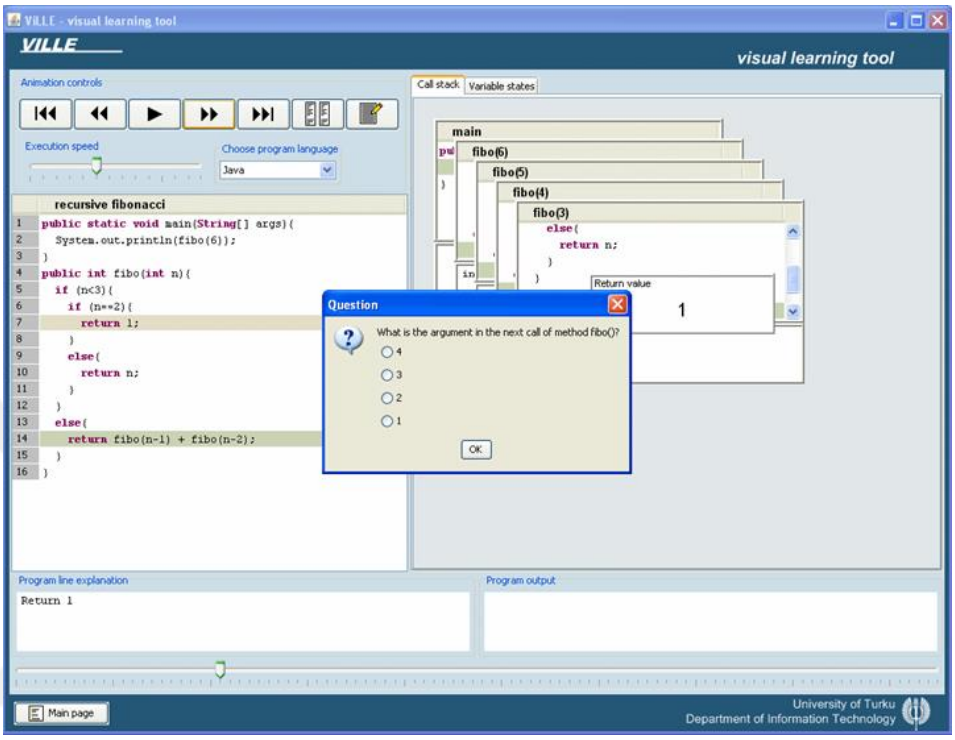
StageCast Creator Yazılımı Ekran Görüntüsü

Ville

Rajala, Laakso, Kaila ve Salakoski (2008), öğrencilerin programlama ile ilgili konuları anlamalarını kolaylaştırmak için Ville adlı bir görselleştirme aracı geliştirmiş ve bu aracı araştırmalarında kullanmışlardır.

Şekil 2.13

Ville Yazılımı Ekran Görüntüsü



Ville, öğrencilere programların çalışmasını inceleme imkânı sunmaktadır. Programın ana özelliği dil bağımsızlığıdır. Yani program, uygulamalarının iki farklı dilde yapılabilmesine ve yeni programlama dillerinin tanımlanmasına olanak vermektedir. Kullanıcı, programı çalıştırma işlemi sırasında değişkenlerin değerlerindeki değişimi ve program çıktısındaki değişimi gözlemleyebilmektedir.

ToonTalk

Bir diğer araç ise çocuklara yönelik tasarlanmış olan ToonTalk'tur. Amaç programlama dilinin yazım kurallarını takılmadan animasyon tarzında hareketlendirilmiş programlama yapmaktır. Programcı, programlamanın soyut kavramlarını kendisine sunulan sanal bir ortamdaki karakterleri basit anlaşılabilir kelimelerle kullanarak hareketlendirebilir. ToonTalk'daki bulunan

her şey görülüp, tutulabilir ve üzerinde değişiklik yapılabilir. Çocuklar karakterlere çeşitli mesajlar atayarak onların bir şeyler yapmasını programlayabilmektedir. Örneğin robotların kutularla iş yapması sağlanabilir, onları kamyonlara yükleyebilir ve hareketlendirilmiş araçları kullanarak bunları kopyalayıp, silip, uzatabilir. Programlamaya yeni başlayan her yaştaki öğrencilerin öğrenme tipleri birbirine yakındır. Yetişkinler ve çocuklar benzer yollarla öğrenir. Bu yüzden ToolTalk sadece çocuklar için değil yetişkin eğitimlerinde de kullanılabilir (Baldwin & Kuljis, 2001). Toontalk'da programlama ortamı bir video oyunu olup kaynak kodlar animasyon ile gösterilmektedir. Toontalk, öğrencilerin bir karakteri kontrol ederek onun programları kurması, çalıştırması, hataları ayıklaması ve değiştirmesini sağlamaktadır (Kahn, 1996).

Şekil 2.14

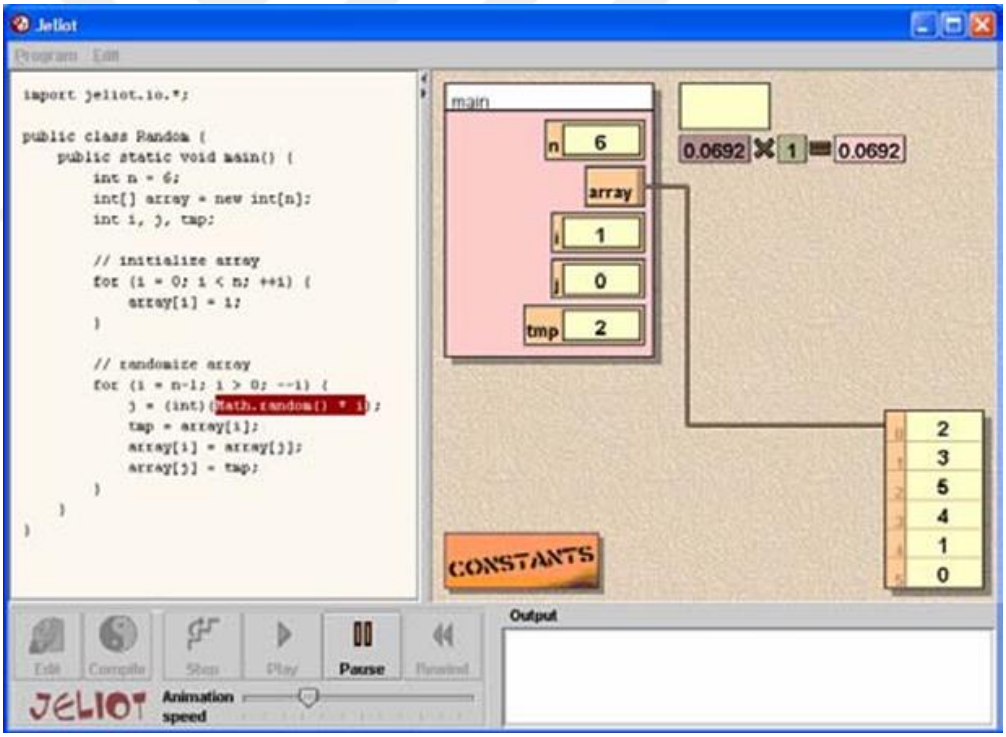
ToonTalk Yazılımı Ekran Görüntüsü

Jeliot 3

Jeliot 3, programlamayı yeni öğrenen öğrencilere yapısal ve nesne yönelimli programlama öğretimi için tasarlanmış bir program görselleştirme aracıdır. Jeliot 3, öğrencilerin kendi programlarını oluşturmalarına katkı sağlamakla kalmayıp aynı zamanda programlarının çalışmasını görsel olarak incelemelerine imkân tanımaktadır (Moreno, Myller, Sutinen & Ben-Ari, 2004).

Şekil 2.15

Jeliot 3 Yazılımı Ekran Görüntüsü



Scratch

Bir başka öğretim yazılımı ise Scratch'tır. 2003 yılında MIT Media Laboratuarı'nda başlatılan Scratch projesi 2007 yılında tüm dünyada yaklaşık 50 dil (Türkçe'de dâhil) desteği sunarak ücretsiz olarak yayınlanmıştır. Yayınlandığı tarihten itibaren Scratch'ın resmi web

sitesinden (*www.scratch.mit.edu*) yaklaşık iki milyondan (şu anda altı milyon) daha fazla kullanıcı tarafından indirilip okullar ve üniversiteler tarafından dağıtılmıştır (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010). Scratch çocukların kendi çoklu-ortam medya tasarımlarını yapmalarını, karşılaştıkları gerçek hayat problemlerine yönelik teknolojiyi kullanarak orijinal çözümler üretebilmelerini ve kendilerini değişik şekillerde ifade ederek 21. yüzyıl üretim becerilerini kazanmalarını amaçlıyor (ScratchÖgren, 2015). Scratch ile kod bloklarını sürükle-bırak yapıp bir araya getirerek grafik, fotoğraf, ses unsurlarının da desteğiyle anlamlı tasarım projeleri ortaya çıkarmak geleneksel programlama dillerinden çok daha kolaydır. Bloklar yazım hatalarını engellemek için birbirlerine uyumlu olarak tasarlanmışlardır. Program çalışırken yeni bloklar eklenebilmekte ve kodların sonuçları ekranda görüntülenebilmektedir (Resnick, 2007).

Şekil 2.16

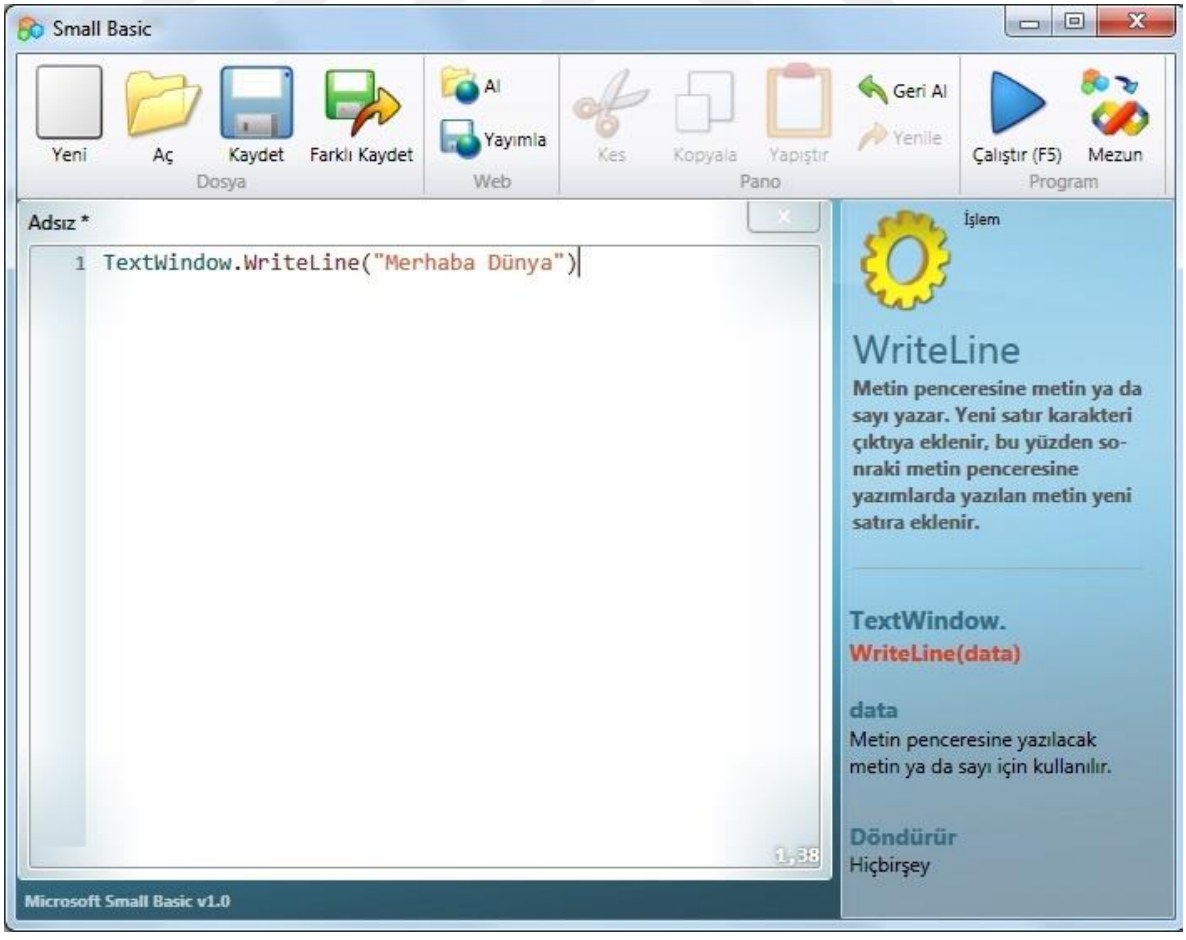
Scratch Yazılımı Ekran Görüntüsü

Microsoft Small Basic

Kod tabanlı ve ücretsiz bir eğitim yazılımı olan Small Basic, Microsoft Visual Basic dilinin olabildiğince basite indirgenmiş mini bir halini temsil etmektedir. Türkçe dil desteği olan Small Basic'in çalışma ortamı, komutlar, kodlar ve menüler çocukların ilgi duyacağı ve kolaylıkla kullanabileceği bir şekilde hazırlanmıştır. Kod alanına yazılan kodlar ise diğer dillerde olduğu gibi ne yapılacağını kestirebildiğiniz yarı İngilizce ifadeler kullanılarak yazılmaktadır. Temel programlama dili prensibinde kodlar satır satır yazılıp yukarıdan aşağıya doğru çalıştırılmaktadır.

Şekil 2.17

Microsoft Small Basic Yazılımı Ekran Görüntüsü

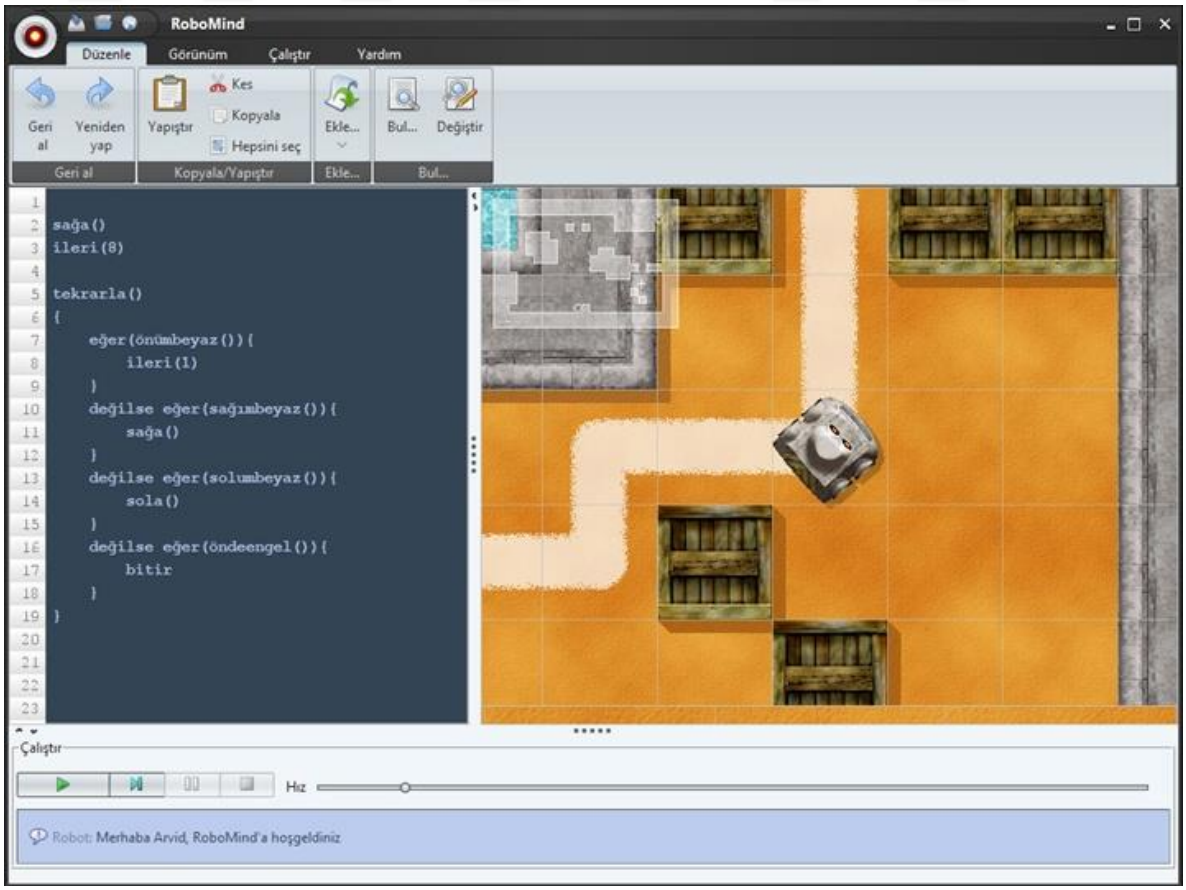


RoboMind

Java kullanılarak Arvid Halma tarafından geliştirilen (Robomind, 2015) yazılımı, bir robotu kontrol ederek temel programlama kavramlarını öğreten bir araçtır. Bu program Logo programlama dili esas alınarak geliştirilmiştir. Ekranın solunda yer alan komut girme alanında, içerisinde Türkçe'nin de yer aldığı dillerde kısa ve öz komutlardan meydana gelen bir dil ile harita üzerindeki robot hareket ettiriliyor. Arayüz dili olarak da Türkçe dil desteğine sahip olan Robomind ile farklı zorluk seviyelerinde haritalar hazırlanarak programlama öğretiminde zihinsel beceriler artırılabilir. Kişisel öğrenim imkânı tanıdığı gibi sahip olduğu öğretim programına göre çevrim içi ücretli olarak grup eğitimlerde düzenlenebiliyor.

Şekil 2.18

Robomind Yazılımı Ekran Görüntüsü

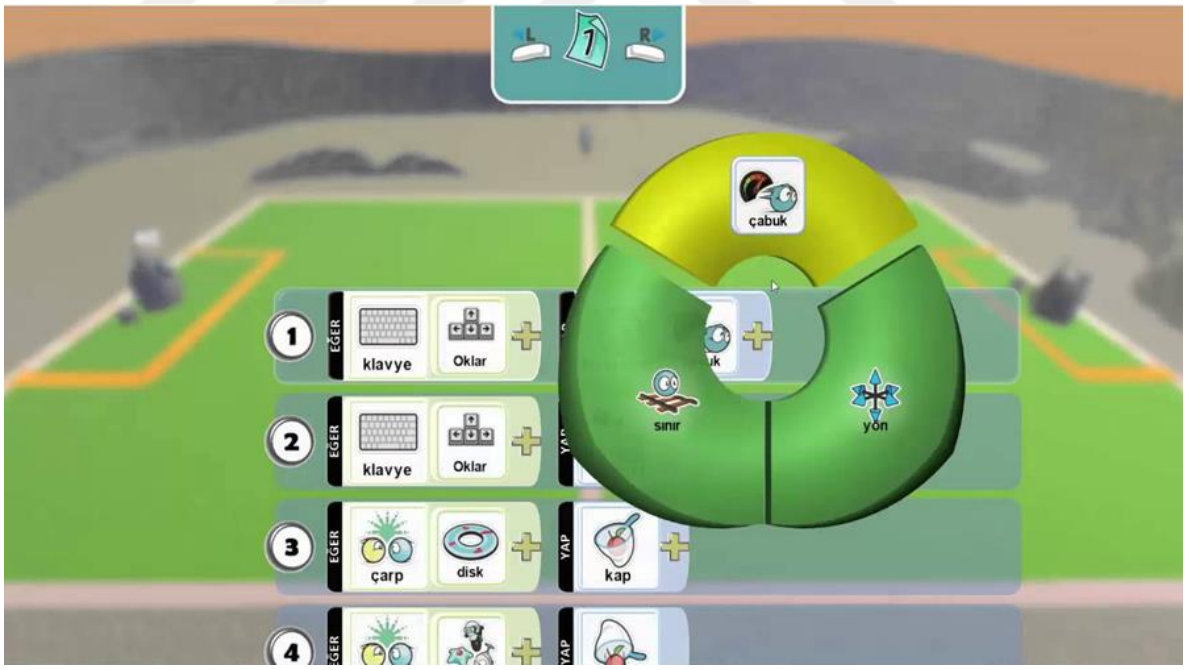


Microsoft Kodu Game Lab

XNA Game Studio ile birlikte Microsoft Research'in iki yılda geliştirdiği bir oyun tasarım yazılımı olan Kodu Game Lab programlama becerisi olmayan çocuklar veya gençler için tasarlanmış kolay kullanımı olan nesne tabanlı görsel bir programlama dilidir. Programlama işlemi kullanıcı ara yüzü sayesinde gerçekleştirilmektedir. Nesne tabanlı ve basit bir arayüz bulunan Kodu yüksek seviye gölge (HLSL) kullanılarak C# dilinde üretilmiştir. Arayüz sayesinde sayfalar arası geçiş, geniş bir yardım menüsü ve diğer özelliklerin tanımı bulunmaktadır (EBA, 2015).

Şekil 2.19

Kodu Game Lab Yazılımı Ekran Görüntüsü



Kodu'da bulunan programlama dili semantik bir yapıya sahiptir; isimler, fiiller ve sıfatlar üzerine kurulu bir yapı sunan dil, topa yaklaşınca yok et gibi basit cümleler kurarak olaylar yaratabilmenizi sağlıyor. Kodu'yu güçlü kılan görsel bir arayüz ve nesneye dayalı programlama

ilkelerine tam uyumlu olmasıdır. Komutlara hiyerarşik bir menünün altında işlevlerini anlatan ikonlarla ulaşılırken, bütün objeler kendi kodlarını içeriyorlar. Yani bir objeye yeni olay tanımlamak istediğinizde bunu direkt olarak o objeyi seçerek kendisine programlıyorsunuz**.** Ücretsiz ve Türkçe dil desteği olan Kodu, oluşturduğunuz sanal dünyanın içereceği obje sayısına ya da bir objenin sahip olabileceği kod miktarına hiç bir sınırlama getirmiyor (Programlar, 2015).

Kodu Game, başlangıçta Xbox 360 platformunda çocuklar için kendi oyunlarını yapabildikleri bir proje olarak sunuldu. Özellikle çocukların programlama alanında basit seviyede eğitimini hedefleyen Kodu, yayınlandığı zamandan itibaren birçok eğitimcinin dikkatini çekmiş durumda ve Amerika'da şu anda 60'a yakın kurumda bir programlama öğretim aracı olarak kullanılıyor (Programlar, 2015).

Sonuç olarak programlama öğretimine yardımcı olmak için yapılmış öğretim araçlarının hazırlanma sebepleri benzerlik gösteriyor. Programlama dillerinin kalıplarını öğrenmedeki zorluklar, ana dildeki eğitim yapmanın daha etkili oluşu, animasyonların dikkat ve ilgi çekmesi, çocuklara hitap edecek bir programlama dilinin olmayışı, her programlama dilinin kendine özgü kurallarının olması ve bunların hepsini bilmenin zorluğu gibi sebepler öğretimde bir materyalin olmasını gerektirmektedir.

2.4.12 Programlama öğretiminde değerlendirme. Programlama becerisinin kazanılması gerek akademik, gerekse profesyonel anlamda ihtiyaç duyulan bir yeterlilik haline gelmiştir. Bu becerinin edinilmesi için kişisel özelliklerle programlama becerisi arasındaki ilişki araştırılarak ve hangi özellikler doğru beceriyi kazandırıyorsa o özelliklerin kişide var olup olmadığını sorgulayarak yapmak mümkündür. Programlama başarısını etkileyen kişisel faktörlerin neler olabileceğini göstermek bu alana yönelik yetişecek olan öğrencilerin tespit edilmesi ve desteklenmesi bakımından çok büyük bir önem taşımaktadır (Erdoğan, 2005).

Colin, Gray, Pavlos ve Athanaios (2005), Nottingham üniversitesinde yürüttükleri bir araştırmada programlama problemlerini bilgisayar tabanlı Course Marker adında bir yazılımla değerlendirme yapmışlardır. Eğiticiler çeşitli değerlendirme sınavlarını seçip öğrencileri bilgisayar üzerinden sınav yapmaktadır. Yazılım, sonuçları ve geri dönütleri öğrencilere hemen döndürmektedir.

Türkiye'de programlama öğretimi bilişim teknolojileri ve yazılım dersi öğretim programı içerisinde verilmektedir. Gerek ders kitabının olmayışı gerekse de belirli bir değerlendirme sisteminin olmayışı Bilişim Teknolojileri öğretmenlerini kendi çözümlerini üretme yoluna sevketmiştir. Bu bağlamda bilgisayar laboratuvarı olmayan okullarda kâğıt üzerinde çoktan seçmeli sınavlar yapılırken, bir laboratuvara sahip olan okullarda ise yine kimileri bilgisayar destekli uygulamalı sınav yaparken kimileri ise proje odaklı ürün oluşturma yoluna giderek programlama becerisi yeterliliklerini değerlendirmektedirler.

2.4.13 Programlama öğretiminde kullanılan yazılımlar üzerine yapılan araştırmalar. Programlama öğretiminde kullanılan yazılımlarla ilgili yapılan çalışmalarda öğretimde bir yazılım kullanılmasının kalıcı öğrenmeye pozitif bir etkisi olduğu, problem çözme ve zihinsel becerileri artırdığı, görselleştirmenin soyut kavramların anlaşılmasına katkıda bulunduğu yönünde sonuçlar çıkmıştır.

Lai ve Repman'nın (1996) yapmış oldukları araştırmanın amacı programlama kavramlarının öğretiminde benzeşimin etkilerini araştırmaktır. İlk olarak katılımcılara programlama dilleri hakkında ne bildikleri, cinsiyet ve yaşları hakkında bir anket uygulanmıştır. Bir hafta sonra hazırlanan dersler bilgisayar laboratuvarında uygulanmıştır. Daha sonra metotları sınayıcı bir uygulama yapılmıştır. Veri analizinden anlaşıldığı üzere bilgisayar programlama dili öğreniminde kavramsal bilgilerin verilmesinde benzeşimin başarıyı artırdığı tespit edilmiştir.

Goldenson (1996), yaptığı çalışmayla "Karel the Robot" programlama eğitimi alan öğrencilerin Pascal programlama diline geçtiklerinde kavramları daha kolay anladıkları ve program yazmada daha başarılı olduklarını belirlemiştir.

Programlama öğretimine yardımcı araçlarda yer alan görsel özelliklerin önemini vurgulamak amacıyla yapılan araştırmaların sonuçları, görsel özelliklerin öğrenci başarısı ve motivasyonunda önemli bir etkisi olduğunu göstermektedir (Ramadan, 2000).

Miyadera, Huang ve Yokoyama'nın (2000) yapmış olduğu bir araştırmada her öğrenci tarafından rahatça kullanılabilecek Java tabanlı bir program görselleştirme aracı tasarlanmıştır. Bu yazılım programcılıkta yeni olan öğrenciler üzerinde denenmiştir. Bu yazılımla gerçekleştirilen öğretim süreci öğrencilerin çeşitli sıralama algoritmalarını öğrenmelerinde yardımcı olmak amacıyla geliştirilmiştir. Öğrenciler üzerinde yapılan deneysel çalışma sonucunda programlama gösterimi ve animasyon teknikleri içeren bu program öğrencilere programın nasıl çalıştığını göstererek programlama becerisi üzerinde olumlu bir etki göstermiştir.

Hundhausen, Douglas ve Stasko (2002), yaptıkları çalışmada görselleştirmenin öğrenme üzerinde istatistiksel açıdan anlamlı pozitif etkisi olduğunu vurgulamışlardır. Ayrıca tek başına görselleştirme kullanımının öğrenmeyi kolaylaştırma konusunda yeterli olmadığını, önemli olanın görselleştirme aracı yardımıyla öğrencileri konuya dâhil etmek olduğunu belirtmişlerdir.

Cooper, Dann ve Pausch'in (2003) yaptıkları bir araştırmada Ithaca College ve Saint Joseph Üniversitesi'nde 2001-2002 eğitim-öğretim yılında programlama bilgisi olmayan 21 öğrenciyle çalışmıştır. Bu öğrencilerden 11'i Alice programını kullanmıştır. Araştırma sonucunda programı kullanan grup, Computer Science 1 dersinde programı kullanmayan gruba göre daha başarılı olmuşlardır.

Moreno, Myller, Sutinen ve Ben-Ari (2004) araştırmalarında Jeliot 3 aracını, Joensuu Üniversitesi'nde programlama dilleri 2 dersinde kullanmışlardır. Araştırmada öğrencilerin Jeliot 3 ile ilgili düşüncelerine yer verilmiştir. Araştırmanın sonucunda Jeliot 3 aracını kullanan öğrencilerin daha başarılı olduğu görülmüştür. Araştırma sonucunda öğrenciler, görselleştirme aracında yer alan dönütleri; koşul cümleleri, döngü ve nesneleri anlamayı kolaylaştırması nedeniyle olumlu bulmuşlardır. Öğrenciler, ayrıca animasyonların yavaş çalıştığı ve aracın gelişmiş dersler için uygun olmadığı yönünde görüş belirtmişlerdir.

Klassen'in (2006) çalışmasına göre, programlama konularının görseller veya animasyonlar yardımı ile sunumu, öğrencilerin motivasyonunu artırmaktadır. Araştırma, programlama derslerinde animasyon ve görsellerin kullanımının, öğrencilerin derse katılımını artırdığını ve öğrencilerin konuları akılda tutmasını kolaylaştırdığını vurgulamaktadır.

Bishop-Clark, Courte ve Howard (2007), yaptıkları çalışmada görselleştirme aracı olarak Alice isimli programlama aracını kullanmışlardır. Araştırmada yer alan öğrenciler, programlama dersi kapsamında 2,5 hafta boyunca Alice ortamında grafiksel programlama uygulamalarına katılmışlardır. Araştırmanın sonucunda programı kullanan öğrencilerin programlama kavramlarını daha iyi anladığı ve programlama ile ilgili kendilerine güvenlerinin arttığı belirtilmiştir.

Peppler ve Kafai (2007), çalışmalarında katılımcılara Scratch programı yardımıyla bilgisayarda video oyunu programlama öğretmişlerdir. Katılımcılar, Scratch yazılımı ile 19 çeşit oyun tasarlamışlardır. Araştırmanın ikinci yılında aynı katılımcılar ile oyun tasarlama çalışmaları devam ettiğinde, katılımcıların ses ve kostüm kullanarak daha karmaşık oyun programlayabildikleri gözlenmiştir. Araştırma sonuçları, ilk ve ikinci yılda geliştirilen projeler karşılaştırıldığında, öğrencilerin ilk yıla göre video oyunu tasarlama konusunda daha fazla bilgi

sahibi olduklarını göstermektedir. Araştırmada Scratch yazılımı kullanılarak video oyunları hazırlamanın, programlama öğrenmeye yardımcı olduğu belirtilmektedir.

Rajala, Laakso, Kaila ve Salakoski (2008), geleneksel yöntemler ile Ville aracının kullanıldığı öğretim yöntemlerini karşılaştırdıkları çalışmalarının sonucunda Ville'nin öğrencilerin temel programlama kavramlarını öğrenmelerini kolaylaştırdığını, ayrıca programlama öğrenmeye yeni başlayan öğrenciler üzerinde daha etkili olduğunu vurgulamaktadırlar.

Fesakis ve Serafeim (2009), bilişim öğretmeni adaylarının Scratch isimli blok programlama dilini, K12 seviyesinde eğitim için mesleki anlamda nasıl algıladığını incelemişlerdir ve Scratch ile çalışmış öğrencilerin, eğitimde blok-programlamanın kullanılmasına olumlu baktığı sonucuna varmışlardır. Yine Scratch kullanarak bu çalışmanın bir benzerini yürüten Choi de aynı sonucu doğrulayan sonuçlar elde etmiştir (Choi, 2012).

Ülkemizde ise, Akçay (2009), çocuk programlama dilinin Türkiye'de Bilgisayar derslerine entegrasyonunu sorgulayan bir araştırma yapmıştır. Araştırmada ilköğretim 4.ve 5. sınıfta okuyan 68 öğrenciye Microsoft firmasına ait Small Basic programı kullandırılmıştır. Araştırma sonucunda öğrenci ve öğretmenlerin programlamaya yönelik algıları incelendiğinde yeni teknolojilerin öğrenci motivasyonlarını olumlu yönde etkilediği tespit edilmiştir.

Calder (2010), Yeni Zelanda'da "Matematiksel düşünmeye bir problem çözme yaklaşımı olarak Scratch programlama dili" tanımıyla yola çıktığı çalışmada 26 öğrenciye tasarım ve kullanım açısından kolay bir programlama dili olan Scratch programını kullandırıp gözlem ve görüşmeler yapmış, öğrencilerin yazdığı yansıma raporlarını incelemiştir. Araştırma sonucunda matematiksel kavramların anlaşılmasında programlama ortamının verimli ve motive edici

olduğunu tespit etmiştir. Calder, önerilerinde matematiksel düşünmeye bir problem çözme yaklaşımı olarak bilgisayar programlama eğitiminin verilmesi gerekliliğini vurgulamıştır.

Prawalpatagool (2010), tarafından 111 10.sınıf öğrencisi arasında Robomind yazılımı kullanılarak yapılan araştırmada programlama öğreniminde başarılarının yükseldiği görülmüştür.

Kaucic ve Asic (2011), Slovenya'da yaptıkları çalışmada 4, 7 ve 9. sınıftan aldıkları toplam 32 öğrenciyle 5 ay boyunca Scratch programlama dili ile araştırma yapmıştır. Araştırma sonucunda programlama dili başlangıç eğitiminin bu şekilde basite indirgenmiş bir ortamda verilmesinin programlama öğrenimine olumlu yönde katkı sağladığı belirlenmiştir.

3. Bölüm

Yöntem

3.1. Araştırma Modeli

Bu araştırma nicel bir araştırma olup, nicel araştırma yönteminde, araştırma evreninin araştırma konusu hakkındaki fikrinin yönü sorgulanmaktadır. Diğer bir deyişle, konu hakkında yoğun bir analiz değil aksine, daha yüzeysel olup, daha çok sayısal veriler saptanmaktadır. Bu anlamda araştırılan konuyla ilgili evreni temsil edecek örneklem grubundan sayısal sonuçlar elde edilmektedir. Araştırmada model (deseni) olarak tekil tarama modeli esas alınmıştır. Değişkenlerin tek tek, tür ya da miktar olarak oluşumlarının belirlenmesi amacı ile yapılan araştırma modellerine tekil tarama modelleri denir. Bu tür bir yaklaşımda, ilgilenilen olay, madde, birey, grup, kurum, konu vb. birim ve duruma ait değişkenler, ayrı ayrı tanıtılmaya çalışılır (Karasar, 2013). Tekil tarama modelli araştırmalarda, daha çok, betimsel istatistik teknikleri gerekli olur. Araştırmacı, en azından, ortalama (medyan), tepe değer (mod), standart sapma, değişkenlik (varyans), dizi genişliği (ranj), frekans dağılımı, normal dağılım, oran, yüzde vb. kavramları bilip uygulayabilmektedir (Karasar, 2013). Genellikle geniş popülasyonlar için uygun olup, araştırmacının uyguladığı anketler ve görüşme planları yaygın veri toplama tekniklerinden ikisidir. İkisi de soruların standardizasyonunu gerektirir (Bozkurt, 2015).

3.2. Çalışma Grubu

Araştırma, Bursa ilinde 2014-2015 eğitim-öğretim yılında tamamı ortaokulda Bilişim Teknolojileri (BT) öğretmeni olarak görev yapan 92 kişi üzerinde yapılmıştır. Bu BT öğretmenlerinin ilçelere göre dağılımı Tablo 3.1'de gösterilmiştir.

Tablo 3.1

BT Öğretmenlerinin İlçelere Göre Frekansları

İlçe	Kişi Sayısı
Büyükorhan	1
Gemlik	5
Gürsu	3
İnegöl	7
İznik	1
Karacabey	5
Keles	3
Kestel	1
Mudanya	4
Mustafa Kemal Paşa	5
Nilüfer	21
Orhaneli	3
Orhangazi	3
Osmangazi	17
Yenişehir	3
Yıldırım	10
Toplam	92

Araştırmaya katılan BT öğretmenlerinin cinsiyet dağılımı ise Tablo 3.2'de gösterilmiştir.

Tablo 3.2

BT Öğretmenlerinin Cinsiyet Frekansları

Cinsiyet	Kişi Sayısı
Kadın	35
Erkek	57
Toplam	92

Araştırmaya katılan BT öğretmenlerinin Milli Eğitim Bakanlığı (MEB)'ndaki kıdem süreleri Tablo 3.3'de gösterilmiştir.

Tablo 3.3

BT Öğretmenlerinin Kıdem Süreleri Frekansları

1-3 yıl	4-6 yıl	7-9 yıl	10-12 yıl	13-15 yıl	16-18 yıl	Toplam
11	26	34	19	1	1	92

BT öğretmenlerinin sahip oldukları programlama bilgisi ise kendilerinden alınan demografik bilgilere göre Tablo 3.4'de gösterilmiştir. Buna göre öğretmenlerden sahip oldukları programlama bilgilerini 1 ile 10 arasında bir derece vermeleri istenmiştir.

Tablo 3.4

BT Öğretmenlerinin Sahip Oldukları Programlama Bilgisi Frekansları

1 derece	2 derece	3 derece	4 derece	5 derece	6 derece	7 derece	8 derece	9 derece	10 derece
-	1	4	6	18	10	28	18	7	-

Araştırmada programlama öğretimi için Scratch 1.4, Small Basic 1.2, Microsoft Kodu Game Lab 1.4.64, Robomind 6.01 yazılımları kullanılmıştır. Bölüm 2'de geçen programlama öğretimi yazılımları başlığında da anlatıldığı gibi bu yazılımların popüler ve birçok ülkede güncel olarak kullanılıyor olmasından ötürü bu yazılımlar seçilmiştir. Ayrıca araştırmacının web üzerinden incelediği kadarıyla Türkiye'de de birçok okuldaki BT öğretmeninin bu yazılımları kullandığı bilinmektedir. Bu görselleştirme araçlarının BT öğretmenleri arasındaki paylaşımı ise Tablo 3.5'de gösterilmiştir. Yazılımlar BT öğretmenlerine rastgele dağıtılmıştır.

Tablo 3.5

Görselleştirme Araçlarının BT Öğretmenlerine Göre Dağılımı

Scratch	Microsoft Small Basic	Microsoft Kodu Game Lab	Robomind	Toplam
21	24	24	23	92

3.3. Veri Toplama Araçları

Araştırmada elde edilen veriler, Deniese Pierotti'nin Nielsen'in (2015a) sezgisel kurallarını esas alarak 2005 yılında yapmış olduğu Sezgisel Kontrol Aracı sayesinde toplanmıştır. Bu araç oldukça detaylıdır. Konu alanı uzmanı eşliğinde araştırma konusuna göre yazılım kullanılabilirliği ve tasarımını inceleyen sorular haricindekiler diğerleri kontrol aracından çıkarılmıştır. Bu şekilde EK-1'de sunulan yeni bir kontrol aracı meydana gelmiştir. Bu yeni sezgisel kontrol aracında BT öğretmenlerinin demografik bilgilerinin alınması dışında programlama öğretiminde kullanılan görselleştirme yazılımlarının kullanılabilirlik ve tasarım kavramları bakımından incelenmesini sağlayan 12 ayrı başlık bulunmaktadır. Bu başlıklar şu şekildedir:

1. Sistem Durumunun Görünürlüğü;

2. Sistem ve Gerçek Dünyanın Eşleşmesi;

3. Kullanıcı Kontrol ve Özgürlüğü;

4. Tutarlılık ve Standartlar;

5.Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma;

6. Hataları Önleme;

7. Hatırlama Yerine Tanıma;

8. Esneklik ve Verimlilik;

9. Estetik ve Sade Tasarım;

10. Yardım ve Dokümantasyon;

11. Yetenekler;

12. Kullanıcı ile Seviyeli Bir İletişim.

Her başlığın altında o başlıkla ilgili bahsedilen kavramın olup olmadığını tespit amacıyla sorular bulunmaktadır. Soruların Evet, Hayır ve Uygulaması Yok şeklinde üç farklı seçeneği vardır. Eğer soruda bahsedilen özellik görselleştirme aracında bulunuyorsa Evet, istenileni karşılamıyorsa Hayır, aranılan özelliğin yazılımda bir karşılığı bulunmuyorsa yani yoksa Uygulaması Yok cevabı verilecektir.

Oluşturulan yeni sezgisel kontrol aracı BT öğretmenlerine Google Forms arayüzünde bir form haline dönüştürülmek suretiyle öğretmenlerin e-posta hesaplarına bağlantı adresi (link) gönderilmiştir.

3.4. Verilerin Toplanması

Araştırmaya katılan BT öğretmenleri sezgisel kontrol aracını 14 Nisan 2015 ve 30 Mayıs 2015 tarihleri arasında doldurmuşlardır. Araştırmaya destek vermeyi kabul eden BT öğretmenlerinin e-postalarına EK-2'de verilen örnek bir e-posta gönderilmiştir. Her BT öğretmeni kendisine e-posta yoluyla gelen programlama öğretiminde kullanılan bir görselleştirme aracını kullanarak onun kullanılabilirlik ve tasarım açısından değerlendirmesini yapmıştır. Bu e-postada, BT öğretmenine hangi görselleştirme aracının verildiği, bu aracın hangi bağlantıdan bilgisayara indirileceği, görselleştirme aracının arayüzünü tanıtan araştırmacı tarafından

hazırlanmış bir YouTube video bağlantısı (Bakınız Tablo 3.6), araştırmanın başlangıç ve bitiş tarihi, bu çalışmanın nasıl ve hangi şartlarda yapılacağını anlatan bir yönerge ve son olarak yukarıda bahsedilen sezgisel kontrol aracının form halindeki bağlantısını içeren detaylı bilgiler sunulmuştur.

Her BT öğretmeni kendisine gelen e-postada yer alan bilgilere göre hareket ederek, verilen görselleştirme aracının paket programını (yazılım) internetten indirip bilgisayarına kurmuştur. Çalışmada BT öğretmenlerinden programlama öğretimindeki uzmanlık bilgisine göre bu yazılımları değerlendirmeleri istenmiştir. Bunun için ilk olarak görselleştirme aracının arayüzünün tanıtıldığı video izlendikten sonra bu yazılımın incelenmesi beklenmiştir. Son olarak ise e-postada verilen sezgisel kontrol aracı bağlantısını internetten açarak inceledikleri yazılıma göre ilgili formu doldurmuşlardır.

Tablo 3.6

YouTube Video Bağlantıları

Görselleştirme Aracı	**YouTube Bağlantı Adresi**	**Video Süresi**
Scratch	https://www.youtube.com/watch?v=z7dRQgl2WII	28:19
Microsoft Small Basic	https://www.youtube.com/watch?v=iwSN6yEdxQY	55:40
Microsoft Kodu Game Lab	https://www.youtube.com/watch?v=tnMlUQKVS7U	37:35
Robomind	https://www.youtube.com/watch?v=zwbBNFcINUo	32:43

3.5. Verilerin Çözümlenmesi

Araştırma için verilen süre bittikten sonra Google Forms platformunda toplanan veriler Microsoft Excel (xlsx) formatında dışarı aktarılmıştır. Girilen verilerin geçerlilikleri konu alanı uzmanıyla tek tek kontrol edilmiştir. Buna göre sezgisel kontrol aracında yer alan ana başlıkların altındaki sorulara verilen Evet, Hayır ve Uygulaması Yok yanıtlarının ayrı ayrı frekans, yüzde ve

ortalamaları hesaplanmıştır. Elde edilen bulgular Bölüm 4’de tablolar halinde gösterilerek yorumlanmıştır.

4. Bölüm

Bulgular

Bu bölümde araştırmadan elde edilen veriler iki farklı analizden geçirilecektir. İlk olarak programlama öğretiminde kullanılan görselleştirme araçlarının kullanılabilirlik ve tasarım açısından barındırdığı sorunlar ve bunların sayısal karşılıkları bulunacak, ikinci olarak bu sorunlara kullanılabilirlik ve tasarım kavramlarından hangisinde karşılaşıldığı tespit edilmeye çalışılacaktır. Bu sayede görselleştirme araçlarının hangi yönü ne ölçüde zayıf ya da kuvvetli olduğu ortaya çıkacağı düşünülmektedir.

4.1. Demografik Bilgilerin Analizi

BT öğretmenleri tarafından programlama öğretiminde görsel araç kullanımı Tablo 4.1'deki gibi çıkmıştır.

Tablo 4.1

BT Öğretmenlerinin Programlama Öğretiminde Görsel Araç Kullanımına Olan Bakışları

Soru – 6	Scratch	Small Basic	Kodu Game	Robomind	Toplam
Çok faydasının dokunacağı kanaatindeyim.	20	19	19	20	78
Kararsızım.	0	1	3	1	5
Yararlı olacağını düşünmüyorum	1	4	2	2	9

Tablo 4.1'e göre programlama öğretiminde görsel bir yazılım kullanmaya %84,7 oranında (78 kişi) olumlu bakılmaktadır.

Kullanılan görsel yazılımların hangi öğretim kademelerinde kullanılması gerektiğine dair görüşler Tablo 4.2'de verilmiştir.

Tablo 4.2

Öğretim Kademelerine Göre Görselleştirme Araçlarının Frekansları

Soru – 7	Scratch	Small Basic	Kodu Game	Robomind	Toplam
İlkokul 3-4	0	0	1	0	1
İlkokul tüm sınıflar	0	1	1	1	3
İlkokul Toplam	1	1	2	1	4 (%4,3)
Ortaokul 5-6	9	1	4	9	23
Ortaokul 7-8	1	8	4	2	15
Ortaokul tüm sınıflar	7	7	8	7	29
Ortaokul Toplam	17	16	16	18	67 (%72,8)
Lise 9-10	0	1	1	2	4
Lise 11-12	0	1	0	0	1
Lise tüm sınıflar	1	3	0	0	4
Lise Toplam	1	5	1	2	9 (%9,7)
İlkokul ve Ortaokul	2	0	3	1	6
Ortaokul ve Lise	1	2	2	1	6
İki Kademe Toplam	3	2	5	2	12 (%13,0)
Genel Toplam	**21**	**24**	**24**	**23**	**92**

Tablo 4.2'ye göre kullanılan bütün yazılımların %72,8 oranında ortaokul kademesine hitap edebileceği yönünde görüş bildirilmiştir.

Görselleştirme araçlarında kullanılabilirlik ve tasarım kavramlarından hangilerine daha çok dikkat edilmesi gerektiğine dair görüşler Tablo 4.3'de verilmiştir.

Tablo 4.3

Görselleştirme Araçlarında Kullanılabilirlik ve Tasarıma Dikkat Etmeye İlişkin Görüşleri

Soru – 8	Scratch	Small Basic	Kodu Game	Robomind	Toplam
Kullanılabilirlik	1	4	1	2	8
Tasarım	1	1	2	0	4
Her ikisi de	19	19	21	21	80

Tablo 4.3'e göre katılımcıların kullanılan bütün yazılımlarda her iki kavramın birlikte göz özüne alınması gerektiği yönünde görüş bildirdiklerini görebiliyoruz.

BT öğretmenleri kullandıkları görselleştirme araçlarının kullanılabilirliğini Tablo 4.4'de değerlendirmişlerdir.

Tablo 4.4

BT Öğretmenlerinin Görselleştirme Araçlarının Kullanılabilirliklerine İlişkin Görüşleri

Soru – 9	Scratch	Small Basic	Kodu Game	Robomind	Toplam
1 (Çok kötü)	0	0	0	0	0
2	0	0	1	0	1
3	3	10	4	2	19
4	16	12	8	20	56
5 (Çok iyi)	2	2	11	1	16

Tablo 4.4'de katılımcılar kullanılabilirlik bakımından %60 oranında 4 derecesinde yoğunlaşmışlardır. 19 kişi bütün yazılımlara 3 derecesini verirken 56 kişi ise 4 derecesini uygun görmüştür. Ayrıca Kodu Game hariç diğer yazılımlar 3 ve 4 derecelerinin toplamında birbirlerine yakın değerler almıştır.

BT öğretmenleri kullandıkları görselleştirme araçlarının tasarımlarını Tablo 4.5'de değerlendirmişlerdir.

Tablo 4.5

BT Öğretmenlerinin Görselleştirme Araçlarının Tasarımlarına İlişkin Görüşleri

Soru – 10	Scratch	Small Basic	Kodu Game	Robomind	Toplam
1 (Çok kötü)	0	0	0	0	0
2	0	0	2	1	3
3	3	10	5	7	25
4	16	11	8	12	47
5 (Çok iyi)	2	3	9	3	17

Tablo 4.5'te katılımcıların %51'i (47 kişi) görselleştirme araçlarının tasarımlarına 4 derecesini, %27,1'i (25 kişi) ise 3 derecesini uygun görmüştür. 3 ve 4 derecelerinin toplamında yine Kodu Game hariç diğerler yazılımlar birbirlerine yakın değerler almıştır.

4.2. Kontrol Aracı Başlıklarına Göre Öne Çıkan Sorunların Analizi

4.2.1. Sistem Durumunun Görünürlüğü. Sezgisel kontrol aracına göre, sistem (görselleştirme aracı) kullanıcıyı sürekli olarak durumunun nasıl gittiği konusunda gerekli zamanlarda gerekli geribildirimi vererek bilgilendirmelidir. BT öğretmenlerinin görüşlerine göre Tablo 4.6'da görünürlük bakımından aşağıdaki verilere ulaşılmıştır.

Tablo 4.6

BT Öğretmenlerinin Sistem Durumu Görünürlüğü Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	66	84	97	98	86,25
Hayır	17	15	12	9	13,25
Uygulaması Yok	22	21	11	8	15,5

Tablo 4.6'ya göre Scratch yazılımı önemli bir farkla Uygulaması Yok ortalamasının (15,5) üstünde bir frekans almıştır. EK-3'e göre Scratch yazılımı bu başlığın 2. ve 3. sorularına sırasıyla 10 ve 9 Uygulaması Yok cevabı almıştır. İlgili sorular şu şekildedir:

1-2 *Her menüdeki bilgilendirmeler, hatırlatmalar ve hata mesajları sistem içerisinde aynı yerde görünmekte midir?*

1-3 *Eğer hata mesajları açılan bir alan içerisinde gösteriliyor ise, kullanıcı, hatalı olan alanı görebilmekte midir?*

4.2.2. Sistem ve Gerçek Dünyanın Eşleşmesi. Sezgisel kontrol aracına göre, sistem içerisinde kullanılan kelimeler, kavramlar, cümleler kullanıcıya tanıdık olmalıdır ve teknik bir dil kullanılmasından kaçınılmalıdır. Bilgi mantıksal sırasında ve doğallığında gösterilerek gerçek dünyada olan şekliyle sunulmalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.7'de gösterilmiştir.

Tablo 4.7

BT Öğretmenlerinin Sistem ve Gerçek Dünyanın Eşleşmesi Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	108	129	120	132	122,25
Hayır	24	23	38	21	26,5
Uygulaması Yok	15	15	10	8	12

Tablo 4.7'ye göre Kodu Game yazılımı 38 Hayır cevabıyla ortalama üstü bir frekans almıştır. Buna göre Kodu Game aracı, bu başlığın 2. ve 6. sorularına sırasıyla 9 ve 12 Hayır cevabı verilmiştir. İlgili sorular şu şekildedir:

2-2 *Eğer görsel bir ipucu olarak kullanılan şekil varsa, bu kültürel geleneklerle eşleşiyor mu?*

2-6 *Sistem kullanıcı jargonunu kullanıp ve bilgisayar jargonundan sakınmakta mıdır?*

4.2.3. Kullanıcı Kontrol ve Özgürlüğü. Sezgisel kontrol aracına göre, kullanıcılar sık sık sistem fonksiyonlarının seçiminde hata yapar ve bu istenmeyen durumda çok detaya girmeden çıkmak için açıkça belirtilmiş bir "acil çıkış"a ihtiyaç duyarlar. Geri alma (undo) ve yeniden yapma (redo) seçenekleri bu amaçla sunulmaktadır. Araçları bu açıdan incelediğimizde Tablo 4.8'de ki sonuçlar ortaya çıkmıştır.

Tablo 4.8

BT Öğretmenlerinin Kullanıcı Kontrol ve Özgürlüğü Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	101	131	129	138	124,75
Hayır	29	24	28	18	24,75
Uygulaması Yok	17	13	11	5	11,5

Tablo 4.8'e göre incelenen yazılımlar içinde Scratch yazılımının istenen özellikleri karşılamada ortalamanın (124,75) altında kaldığı görülmektedir. Scratch, bu başlığın 2. ve 3. sorularına sırasıyla 10 Uygulaması Yok ve 11 Hayır yanıtı alarak kullanıcı kontrol ve özgürlüğünde istenilen özellikleri yeteri kadar verememiştir. İlgili sorular aşağıdaki gibidir.

3-2 *Açılır pencere olan sistemlerde, kullanıcılar için pencereler arası geçiş kolay mıdır?*

3-3 *Herhangi bir işlem hareketi, veri girişi veya birçok diğer işlem hareketleri için "geri al" fonksiyonu bulunmakta mıdır?*

4.2.4. Tutarlılık ve Standartlar. Sezgisel kontrol aracına göre, kullanıcılar, farklı kelime, durum ve hareketlerin aynı şeyi ifade edip etmediğini bilmek zorunda değildirler. Ortak bir tutarlılık tüm sistemde geçerli olmalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.9'da sunulmuştur.

Tablo 4.9

BT Öğretmenlerinin Tutarlılık ve Standartları Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	161	187	180	183	177,75
Hayır	36	43	52	49	45
Uygulaması Yok	24	28	24	12	22

Tablo 4.9'a göre tutarlılık bakımından Kodu Game yazılımı Hayır ortalamasının (45) üstüne çıkmıştır. Detaylara bakıldığında Kodu Game'de bu başlığın 5. ve 6. sorularına sırasıyla 11 ve 10 Hayır yanıtı verilmiştir. Ayrıca Small Basic yazılımı da 28 ortalamayla Uygulaması Yok ortalamasının üstündedir. Bunda da 6. soruya 10 Uygulaması Yok yanıtı verilmiştir. İlgili sorular şu şekildedir:

4-5 *Menüler dikey olarak sunulmuş mudur?*

4-6 *Eğer "çıkış" bir menü seçeneği ise listenin her zaman en altında yer almakta mıdır?*

4.2.5. Kullanıcıların Hataları Tanımasına, Onları Belirlemesine ve Önlemesine Yardımcı Olma.

Sezgisel kontrol aracına göre, hata geri dönütleri, sade bir dilde (kodsuz) olmalı, sorunu açıklamalı ve yapıcı çözüm önerisi sunmalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.10'de verilmiştir.

Tablo 4.10

BT Öğretmenlerinin Hata Tanıma ve Önleme Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	23	95	88	108	78,5
Hayır	44	56	20	43	40,75
Uygulaması Yok	80	17	60	10	41,75

Tablo 4.10'a göre Scratch yazılımı Evet seçeneğinde ortalama altı kalıp Uygulaması Yok seçeneğinde ortalama üstüne (2 katı) çıkmıştır. EK-3'e bu başlığın 2,3,4,5,6. ve 7. sorularına sırasıyla 9,11,13,14,13,12 Uygulaması Yok yanıtı verilmiştir. Diğer taraftan Small Basic yazılımı ise Hayır seçeneğinde ortalama üstüne çıkmıştır. Burada da 1,4 ve 7. sorulara sırasıyla 18,12 ve 13 Hayır yanıtı verilmiştir. Kodu Game yazılımı ise Uygulaması Yok seçeneğinde 60 frekansla ortalama üstünde olduğu görülüyor. Bu yazılım için 4,6 ve 7. sorulara sırasıyla 12,11 ve 12 Uygulaması Yok cevabı verilmiştir. İlgili sorular şu şekildedir:

5-1 *Bir hatayı bildirmek için ses kullanılıyor mu?*

5-2 *Hatırlatıcılar, işlemin kullanıcının kontrolünde olduğunu ima etmekte midir?*

5-3 *Hatırlatıcılar, kısa, net ve anlaşılır mı?*

5-4 *Hata mesajları kullanıcının değil, sistemin sorumlu olduğunu belirtecek şekilde ifade edilmekte midir?*

5-5 *Hata mesajları gramer açısından doğru mudur?*

5-6 *Hata mesajları sorunun nedeni hakkında bilgiler sunuyor mu?*

5-7 *Hata mesajları, kullanıcıya hatayı düzeltmek için hangi hareketi yapması gerektiğini belirtiyor mu?*

4.2.6. Hataları Önleme. Sezgisel kontrol aracına göre, hata mesajları dikkatli ve birinci aşamada oluşan problemleri önleyen bir yapıda olmalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.11'de verilmiştir.

Tablo 4.11

BT Öğretmenlerinin Hataları Önleme Boyutuna İlişkin Görüşleri

Seçenekler	**Scratch**	**Small Basic**	**Kodu Game**	**Robomind**	Ortalama
Evet	46	59	62	59	56,5
Hayır	39	42	32	50	40,75
Uygulaması Yok	22	19	26	6	18,25

Tablo 4.11'de Robomind yazılımı Hayır seçeneğinde ortalamanın (40,75) üstündedir. EK-3'e bakıldığında bu başlığın 2,3. ve 5. sorularına sırasıyla 10,18 ve 11 Hayır yanıtı verilmiştir. Diğer taraftan Kodu Game yazılımı ise Uygulaması Yok seçeneğinde ortalama üzerindedir. Aynı sorulara bu seçenekte sırasıyla 10,13 ve 10 Uygulaması Yok cevabı verilmiştir. İlgili sorular aşağıdaki şekildedir.

6-2 *Veri girişleri mümkün olduğu kadarıyla büyük ve küçük harfe duyarlı olarak girilebiliyor mu?*

6-3 *Ciddi sonuçları olabilecek gösterge/düğmeler ulaşılması zor bir yerde midir?*

6-5 *Sistem, kullanıcıları sonuçları ciddi ve yıkıcı olabilecek hatalar yapmak üzere iken onları uyarıyor mu?*

4.2.7. Hatırlama Yerine Tanıma. Sezgisel kontrol aracına göre, nesne, hareket ve seçenekler görünür olmalıdır. Kullanıcı, sistem boyunca kullanması gereken bilgiyi hatırlamak zorunda kalmamalıdır. Hatırlatma/bilgilendirme/açıklama bölümleri kullanıcının kolaylıkla ulaşabileceği şekilde sistemde yer almalıdırlar. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.12'de sunulmuştur.

Tablo 4.12

BT Öğretmenlerinin Hatırlama Yerine Tanıma Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	134	151	164	149	149,5
Hayır	23	27	23	28	25,25
Uygulaması Yok	11	14	5	7	9,25

Tablo 4.12'ye göre Small Basic yazılımı Uygulaması Yok seçeneğinde ortalama üzerindedir. Detaylara bakıldığında bu başlığın 6. ve 7. sorularına 4 Uygulaması Yok yanıtı verilmiştir. İlgili sorular aşağıdaki şekildedir:

7-6 *Anlamlı grupları belirlemek için çerçeveler kullanılmış mıdır?*

7-7 *İlişkili elemanları gruplamak için aynı renkler kullanılmış mıdır?*

4.2.8. Esneklik ve Verimlilik. Sezgisel kontrol aracına göre, acemi kullanıcılar tarafından fark edilemeyen hızlandırıcılar, uzman kullanıcıların sistemle etkileşimini arttırabilir. Bu nedenle sistem hem deneyimli hem deneyimsiz kullanıcılara hitap edecek şekilde tasarımlanmalıdır. Standart kullanıcılardan ziyade farklı (fiziksel, bilişsel yetenek, kültür, dil, vs

bakımlarından) kullanıcıların sisteme alternatif ulaşma ve işlem yapabilme olanağı sağlanabilmelidir. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.13'de verilmiştir.

Tablo 4.13

BT Öğretmenlerinin Esneklik ve Verimlilik Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	29	46	51	55	45,25
Hayır	28	34	31	34	31,75
Uygulaması Yok	31	16	14	3	16

Tablo 4.13'de incelenen yazılımlardan Scratch Uygulaması Yok seçeneğinde ortalama üstündedir. Buna göre bu başlığın 1. sorusuna 10 Uygulaması Yok yanıtı verilmiştir. İlgili soru aşağıdaki gibidir:

8-1 *Eğer sistem acemi ve profesyonel kullanıcıların ikisini de destekliyor ise detaylı hata mesajları farklı düzeylerde sağlanmış mıdır?*

4.2.9. Estetik ve Sade Tasarım. Sezgisel kontrol aracına göre, sistem alakasız ve az ihtiyaç duyulan bilgiyi içermemelidir. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.14'de verilmiştir.

Tablo 4.14

BT Öğretmenlerinin Estetik ve Sade Tasarım Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	100	116	119	116	112,75
Hayır	22	16	18	17	18,25
Uygulaması Yok	4	12	7	5	7

Tablo 4.14'de Scratch yazılımı Hayır seçeneğinde ortalama üzerindedir. Ayrıntılı bakıldığında bu başlığın 6.sorusuna 10 Hayır yanıtı verilmiştir. İlgili soru aşağıdaki gibidir:

9-6 *Sistem, veri girmek için kutu dışında açılır kutu, radyo butonu gibi başka seçenekler sunuyor mu?*

4.2.10. Yardım ve Dokümantasyon. Sezgisel kontrol aracına göre, sistemi kullanmak için belgeleme olmaması iyi olacağı halde, gerekli durumlarda yardım ve belgeleme sağlanması önerilmektedir. Bu tür yardım bilgileri taranması kolay, kullanıcı görevlerim açıklayıcı ve kısa olmalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.15'de sunulmuştur.

Tablo 4.15

BT Öğretmenlerinin Yardım ve Dokümantasyon Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	81	55	97	107	85
Hayır	26	42	20	18	26,5
Uygulaması Yok	7	34	14	2	14,25

Tablo 4.15'e göre Small Basic yazılımı Hayır ve Uygulaması Yok seçeneklerinde ortalama üzerinde çıkmıştır. Hayır seçeneğinde 3. ve 5. sorulara sırasıyla 10,17 Hayır, Uygulaması Yok seçeneğinde 1. ve 6. sorulara sırasıyla 9,12 Uygulaması Yok yanıtı verilmiştir. İlgili sorular aşağıdaki gibidir.

10-1 *Çevrim içi bilgilendirmeler (online yardım gibi) görsel ola-rak ayırt edilebilir durumda mıdır?*

10-3 *Yardım işlevi görünür mü? Örneğin YARDIM adında bir düğme veya özel bir menü mevcut mudur?*

10-5 *Kullanıcı, var olan detay seviyesini değiştirebiliyor mu?*

10-6 *Yardım sistemine kolayca ulaşılabiliyor mu veya bu bölümden kolayca çıkılabiliyor mu?*

4.2.11. Yetenekler. Sezgisel kontrol aracına göre, sistem kullanıcının yeteneklerinin ve uzmanlığının gelişmesine ve fazlalaşmasına olanak sağlamalıdır. Bu maddeye ilişkin öğretmen görüşleri Tablo 4.16'da sunulmuştur.

Tablo 4.16

BT Öğretmenlerinin Yetenekler Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	107	109	114	125	113,75
Hayır	14	25	21	9	17,25
Uygulaması Yok	5	10	9	4	7

Tablo 4.16'ya göre Small Basic yazılımı Hayır ve Uygulaması Yok seçeneklerinde ortalama üzerinde çıkmıştır. Hayır seçeneğinde 4. ve 5. sorulara sırasıyla 6,8 Hayır, Uygulaması Yok seçeneğinde 4. soruya 4 Uygulaması Yok yanıtı verilmiştir. İlgili sorular aşağıdaki gibidir.

11-4 *Kullanıcılar, bir alan içerisinde ileri ve geri hareket edebiliyorlar mı?*

11-5 *Fonksiyonelliği desteklemeye yetecek kadar sayıda, ancak aramayı ve bulmayı zorlaştırmayacak kadar yeterli işlev komutu (ekranda) var mı?*

4.2.12. Kullanıcı ile Seviyeli İletişim. Sezgisel kontrol aracına göre, kullanıcının sistemle etkileşimi onun iş yaşamındaki kaliteyi arttırmalıdır. Kullanıcıya saygılı davranılmak ve tasarım görünüş ve işlemesi bakımından memnun edici olmalıdır. Bu başlığa göre verilen cevaplar Tablo 4.17'de sunulmuştur.

Tablo 4.17

BT Öğretmenlerinin Kullanıcı ile Seviyeli Bir İletişim Boyutuna İlişkin Görüşleri

Seçenekler	Scratch	Small Basic	Kodu Game	Robomind	Ortalama
Evet	105	128	121	121	118,75
Hayır	17	13	22	16	17
Uygulaması Yok	4	3	2	1	2,5

Tablo 4.17'e göre Kodu Game Hayır seçeneğinde ortalama üzerindedir. EK-3'e bakıldığında bu başlığın 1.sorusuna 5 Hayır yanıtı verilmiştir. İlgili soru aşağıdaki gibidir:

12-1 *Her bir ikon (simge), bir ikon grubunun uyumlu bir öğesi midir?*

4.3. Görselleştirme Araçlarının Sahip Olduğu Sorun Sayılarına Göre Analizi

Sezgisel kontrol aracı başlıklarına göre yapılan yukarıdaki ayrı ayrı analizler sonucu öne çıkan ve farklılık gösteren (problemlerin/sorunların oluştuğu) ilgili sorular ve Hayır/Uygulaması Yok seçeneklerine göre ne kadar yanıt aldıklarına dair tablo EK-4'te sunulmuştur. Bu tablodaki başlıklara göre görselleştirme araçlarının aldıkları Hayır ve Uygulaması Yok yanıtları toplanarak Tablo 4.18'de sunulmuştur.

Tablo 4.18

Kontrol Aracı Başlıklarına Göre Öne Çıkan Problemlerin Frekansları

		1. Sistem Durumunun Görünürlüğü	2. Sistem ve Gerçek Dünyanın Eşleşmesi	3. Kullanıcı Kontrol ve Özgürlüğü	4. Tutarlılık ve Standartlar	5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma	6. Hataları Önleme	7. Hatırlama Yerine Tanıma	8. Esneklik ve Verimlilik	9. Estetik ve Sade Tasarım	10. Yardım ve Dokümantasyon	11. Yetenekler	12. Kullanıcı ile Seviyeli Bir İletişim	Yanıtların Toplamı
	Toplam	**19**	**21**	**21**	**31**	**150**	**72**	**8**	**10**	**10**	**48**	**18**	**15**	**423**
Robomind	Hayır						39						10	49
	U.Yok													
Kodu Game	Hayır		21		21		13						5	60
	U.Yok					35	20							55
Small Basic	Hayır					43					27	14		84
	U.Yok				10			8			21	4		43
Scratch	Hayır			11						10				21
	U.Yok	19		10		72			10					111

Tablo 4.18'e göre bütün yazılımlar için kontrol aracı başlıklarından *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlığında en çok sorunun (150) oluştuğunu görmekteyiz. Bunu sırasıyla *Hataları Önleme* (72), *Yardım ve Dökümantasyon* (48) ve *Tutarlılıklar ve Standart* (31) başlıklarında sorunlar bulunmuştur.

Yazılımlar bazında tabloya göre ilk olarak, en çok soruna (132) sahip Scratch yazılımı *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlığında 72, *Sistem Durumunun Görünürlüğü* başlığında 19 önemli probleme sahip olduğu görülmüştür.

Toplam 127 sorunla ikinci olan Small Basic yazılımı, sırayla *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlıkta toplam 43, *Yardım ve Dokümantasyon* başlığında toplam 48, *Yetenekler* başlığında 18 önemli problemi barındırdığı ortaya çıkmıştır.

Üçüncü olarak 115 sorun bulunan Kodu Game yazılımı sırayla *Sistem ve Gerçek Dünyanın Eşleşmesi* ve *Tutarlılık ve Standartlar* başlığında 21, *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlığında 35, *Hataları Önleme* başlığında 33 problemi bulunduğu tespit edilmiştir.

Son olarak 59 sorun bulunan Robomind yazılımı sırayla *Hataları Önleme* başlığında 39, *Kullanıcı ile Seviyeli Bir İletişim* başlığında 10 problem bulunmuştur.

Tablo 4.18 tersinden okunduğunda kullanılabilirlik ve tasarım sorunlarına göre bir sıralama yapılırsa Tablo 4.19'da ki gibi bir sonuç ortaya çıkacaktır.

Tablo 4.19

Problem Sayısına Göre Görselleştirme Araçlarının Frekans ve Yüzdeleri

Sıralama	Görselleştirme Aracı	Sorun Sayısı	%
1.	Robomind	49	%11,5
2.	Kodu Game	115	%27,1
3.	Small Basic	127	%30,0
4.	Scratch	132	%31,2

Görüldüğü gibi Robomind yazılımı en az probleme sahipken Scratch ise en çok problemi taşıyan yazılım olduğu tespit edilmiştir.

4.4. Kullanılabilirlik Ve Tasarım Problemleri Analizi

Bu başlıkta kontrol aracında bulunan kriterlerin kullanılabilirlik ve tasarım kavramlarından hangilerine karşılık geldiğine bakarak görselleştirme araçlarının yukarıdaki sorun sayılarına göre yapılan analizin kavram boyutu incelenecektir. Diğer bir deyişle, sorun çıkan ve farklılık gösteren soruların hangi kavramla ilgili olduğu bulunup buna göre genel anlamda yazılımların hangi yönünün zayıf veya kuvvetli olduğu değerlendirilecektir. Bunun için konu alanı uzmanının vermiş olduğu destekle oluşturulan EK-5 ile EK-4'ün kesişimi sağlanarak Tablo 4.20'ye göre analiz yapılacaktır.

Tablo 4.20

Görselleştirme Araçlarının Sorunlarının Kavram Bakımından Frekansları

Problem Barındıran Sorular	Kavramların Bulunuşu			Toplam Problem Sayısı (Hayır + Uygulaması Yok)			
	Kullanılabilirlik	Tasarım	Her ikisi	Scratch	Small Basic	Kodu Game	Robomind
1-2			✓	10			
1-3	✓			9			
2-2		✓				9	
2-6	✓					12	
3-2	✓			10			
3-3			✓	11			
4-5		✓				11	
4-6		✓			10	10	
5-1		✓			18		
5-2	✓			9			
5-3	✓			11			
5-4		✓		13	12	12	
5-5		✓		14			
5-6		✓		13		11	
5-7			✓	12	13	12	
6-2			✓			10	10
6-3			✓			13	18
6-5	✓					10	11
7-6		✓			4		
7-7		✓			4		
8-1			✓	10			
9-6		✓		10			
10-1		✓			9		
10-3		✓			10		
10-5			✓		17		
10-6			✓		12		
11-4			✓		10		
11-5			✓		8		
12-1			✓			5	
12-6		✓					10

Tablo 4.20'ye ilk bakışta en çok öne çıkan soruların beşinci başlık yani *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlığında biriktiğini görebiliyoruz. Ayrıca görselleştirme yazılımlarında varolan sorunların tasarım kavramında (13) en çok olduğu görülüyor. Onu her ikisi (11) takip ederken sonuncu olarak kullanılabilirlik (6) kavramı geliyor. Bunu sorun sayılarının toplamına göre Tablo 4.21'den de rahatlıkla görebiliriz.

Tablo 4.21

Görselleştirme Araçlarının Problem Sayılarının Kavramlara Göre Frekansları

	Kullanılabilirlik	Tasarım	Her ikisi
Robomind	11	10	28
Kodu Game	22	53	40
Scratch	39	50	43
Small Basic	0	67	60
Toplam	72	180	171

Tablo 4.21'e göre kullanılabilirlik ve tasarıma (her ikisi) göre en az sorunu olan birinci yazılım Robomind (%16,3) olurken, onu Kodu Game (%23,3) ve Scratch (%25,1) takip etmiştir. En çok sorun bu analizde de yine Small Basic (%35,0) yazılımında bulunmuştur.

Tasarım bakımından en az sorunu olan birinci yazılım Robomind (%5,5), ikinci Kodu Game (%29,4), üçüncü Scratch (%27,7), sonuncu olarak Small Basic (%37,2) çıkmıştır.

Kullanılabilirlik bakımından hiç sorunu bulunmayan yazılım Small Basic olurken ikinci Robomind (%15,2), üçüncü Kodu Game (%30,5) ve sonuncu olarak Scratch (%54,1) çıkmıştır.

Kontrol aracı başlıklarının kullanılabilirlik ve tasarım kavramlarına göre dağılımı ise Tablo 4.22'de gösterilmiştir.

Tablo 4.22

Kontrol Aracı Başlıklarına Göre Kavramların Frekansları

Kontrol Aracı Başlıkları	Kullanılabilirlik	Tasarım	Her ikisi
1. Sistem Durumunun Görünürlüğü	9	-	10
2. Sistem ve Gerçek Dünyanın Eşleşmesi	12	9	-
3. Kullanıcı Kontrol ve Özgürlüğü	10	-	11
4. Tutarlılık ve Standartlar	-	31	-
5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma	20	93	37
6. Hataları Önleme	21	-	51
7. Hatırlama Yerine Tanıma	-	8	-
8. Esneklik ve Verimlilik	-	-	10
9. Estetik ve Sade Tasarım	-	10	-
10. Yardım ve Dokümantasyon	-	19	29
11. Yetenekler	-	-	18
12. Kullanıcı ile Seviyeli Bir İletişim	-	10	5
Toplam	72	180	171

Tablo 4.22'ye göre görselleştirme araçlarında kullanılabilirlik boyutunda en çok sorun *Hataları Önleme* başlığında (%29,1), tasarım boyutunda *Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma* başlığında (%51,6), her iki kavram içinde *Hataları Önleme* başlığında (%29,8) ortak sorunlar bulunmuştur.

Bu bölümde programlama öğretiminde kullanılan görselleştirme yazılımlarının kontrol aracına göre öne çıkan kullanılabilirlik ve tasarım sorunları, bu sorunların sayısal ve kavramsal

olarak analizleri yapılmıştır. Bir sonraki bölümde bulunan bu sorunlar için çözüm ve öneriler verilecektir.

5. Bölüm

Tartışma ve Öneriler

Bu bölümde elde edilen verilere göre yapılan analizler ışığında programlama öğretiminde görselleştirme aracı olarak kullanılabilecek yazılımların kullanılabilirlik ve tasarım açısından değerlendirmesine a) kontrol aracı başlıklarına; b) görselleştirme yazılımlarına; c) kavram boyutuna göre çözümler ve öneriler verilmeye çalışılmıştır.

Tartışma

Araştırmada programlama öğretiminde görsel bir yazılım kullanmaya olumlu (%84,7) bakılmaktadır (Bknz Tablo 4.1). Bu sonuç yapılan araştırmaların (Gülmez, 2009; Baldwin & Kuljis, 2001; Crews & Ziegler, 1998) sonuçlarıyla da örtüşmektedir. Bu durumda, programlama öğretiminde görsellik kullanımının ilköğretim öğrencileri için pozitif etkiler bırakacağı düşünülebilir.

Araştırmada Tablo 4.2'ye göre kullanılan görsel yazılımların genel itibariyle (%72,8) ortaokul kademelerine uygun olduğu sonucu çıkmıştır. Diğer taraftan Small Basic yazılımı daha çok ortaokula hitap ederken %9,7 oranında lise kademesinde de kullanılabileceği gibi bir veri de elde edilmiştir. Buna göre Small Basic'in kod tabanlı yazım düzenine ve arayüze sahip olması ve bu haliyle profesyonel yazılım dillerine yakın durduğu düşünülerek bu sonuç çıkmış olabilir.

Demografik bilgiler başlığında BT öğretmenlerine göre Kodu Game yazılımı hariç diğerlerinin genel itibariyle kullanılabilir ve tasarımlarının da iyi olduğu sonucuna varılmıştır. Bu sonuç, henüz sezgisel kontrol aracı sorularını yanıtlamadan, BT öğretmenlerinin mevcut uzmanlık seviyesi ve kişisel görüşleri alınarak ortaya çıkmıştır. Sezgisel kontrol aracına göre detaylı değerlendirmeler aşağıda sunulacaktır.

Araştırmaya göre;

a) Sezgisel kontrol aracında bulunan başlıklara bakılarak yazılımlarda tespit edilen problemlere bakacak olursak:

Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma başlığında (150 sorun bulunmuştur) yazılımlar hata bildiriminde ses kullanımı, hata mesajlarında kullanıcının değil sistemin sorumlu olduğu, hata mesajlarında gramer sorunları, hatanın neyden kaynaklandığı ve oluşan hatayı düzeltmek için neler yapılması gerektiği konusunda kullanıcıların ihtiyaçlarını karşılayamamaktadır. Kısacası yazılımlarda hata yönetimi bağlamında birtakım problemler bulunmaktadır. Ayrıca oluşan bu problemlerin tasarım kavramı alanında yoğunlaştığını da görmekteyiz. Scratch yazılımı bu başlıkta en çok soruna sahip yazılım olarak göze çarpmaktadır. Scratch yazılımı incelendiğinde oluşan hata mesajlarında sorunun nedeni ve bundan sonra ne yapılması gerektiğini belirten bir ifade bulunmamaktadır. Ayrıca ses kullanarak hata bildirimi de sunulmamaktadır. Gramer açısından ise Scratch'ın Türkçe sürümünde ufak tefek problemler olduğu görülmektedir. "Programdan çıkmadan önce değişiklikleri kaydetmek ister misin?" gibi cana yakın sempatik bir soruya "Kaydedin", "Kaydetme", "İptal" gibi hem kendi aralarında hem de soruya göre çokta uyumlu olmayan çevirileri görmekteyiz. Small Basic yazılımı da ses ile hata bildirimi vermezken hata mesajlarında teknik bir dil kullanılmaktadır ve daha sonrasında yapılması gerekenleri belirtmemektedir. Kodu Game yazılımı ise oluşan hatanın sistem yerine kullanıcı kaynaklı olduğuna dair hata mesajları vermektedir. Bu hataların nasıl düzeltileceğine yönelik herhangi bir yönergede sunulmamaktadır.

Hataları Önleme başlığında (72 sorun bulunmuştur) görselleştirme yazılımlarında veri girişi esnasında büyük/küçük harf duyarlılığının olmayışı, çalıştırılınca sonuçları itibariyle ciddi sorunlar çıkarabilecek düğmeler/göstergelerin kolay ulaşılabilecek yerlerde olması ve

oluşabilecek bu olaylar öncesi kullanıcıların uyarılmaması gibi problemler bulunmuştur. Kodu Game yazılımı tasarım itibariyle tak-çıkar (hot-plug) mantığına göre yapıldığı için kullanıcıdan oyun yapımı sırasında herhangi bir veri girişine imkân vermemektedir. Ayrıca yapılan oyun kurgusunun sonuçlarının ne olabileceği hakkında kullanıcıları uyarmıyor, deneme yanılma yoluyla kullanıcının bulması gerekiyor. Robomind yazılımında ise haritadaki nesneyi (tank) hareket ettiren komut ya da komut bloklarının küçük harfle yazılması zorunludur. Bu şekilde kullanıcı sınırlanmış olmaktadır. Diğer taraftan Robomind girilen komutlar sayesinde oluşabilecek sonsuz döngüler ya da bölünebilme hatalarında programı çalıştırmadan önce sonucun ne olabileceği hakkında kullanıcıya herhangi bir uyarıda bulunmamaktadır.

Yardım ve Dökümantasyon başlığında (48 sorun bulunmuştur) gerekli durumlarda yardım ve belgeleme olması beklenen yazılımlardan Small Basic'in bu ihtiyacı karşılamaktan uzak olduğunu görüyoruz. Tablo 4.15'e göre Small Basic yazılımı incelendiğinde sorularda da beklendiği gibi bir yardım menüsü ya da aynı işleve gören bir düğme bulunmuyor. Ancak ekranın sağ tarafında yer alan alanda kod alanında yazılan komutlara göre o komutun anlamı ve nasıl kullanıldığına dair minik örneklerin sunulduğu bir yardım metodu bulunmaktadır. Sunulan bu yardım işlevinde detay seviyeleri de bulunmamaktadır. Diğer yazılımlarda ise kullanıcıya yön gösterebilecek bir yardım menüsü var olup farklı seviyelerde bilgiler mevcuttur.

Tutarlılık ve Standartlar başlığında (31 sorun bulunmuştur) Kodu Game ve Small Basic yazılımlarının sistem boyunca ortak bir tutarlılığın geçerli olmadığını görüyoruz. Kodu Game yazılımında açılış ekranında yer alan menü dikey iken programlama işlemi esnasında yatay bir menü tasarımı kullanıcıya sunulmaktadır. Yine "çıkış" seçeneği ise yatay menüde en solda ya da "ESC" tuşu ile sağlanmaktadır. Robomind yazılımı ise klasik dikey menü yerine menüyü Ribon tarzı simgeli gruplar halinde kullanıcıya vermektedir.

Kullanıcı Kontrol ve Özgürlüğü başlığında (21 sorun bulunmuştur) yazılımlarda kullanıcıların hata yaptıklarında veya hatayı geri alma işlemlerinde kullanıcıya tam bir özgürlük sunulmadığını görüyoruz. Buna göre Scratch yazılımı incelendiğinde programlama yapılırken yapılan işlemleri geri alma ve yineleme işlemlerinin neredeyse olmadığını belli başlı temel işlemler (kaydetme, silme gibi) için kullanılabildiğini görüyoruz. Blok tabanlı bir görselleştirme aracında geri alma ve yinelemenin çok kısıtlı olması kullanıcı bu bakımdan yormaktadır. Ayrıca "Motor Blokları Göster/Gizle" gibi nadir kullanılan görevleri hatırlamak Scratch'da mümkün gözükmemektedir.

Sistem ve Gerçek Dünyanın Eşleşmesi başlığında (21 sorun bulunmuştur) Kodu Game yazılımında sistem içerisinde kullanılan kelimeler, kavramlar, cümleler kullanıcıya pekte tanıdık gelmediği sonucuna varıyoruz. Buna göre Kodu Game'in Türkçe arayüzünde direkt bir çeviri yapılıp "Yeni Dünya", "Dünya Yükle", "Benim Dünyalarım" gibi ilk bakışta garip gelebilecek bir bilgisayar üslubu kullanıcıyı karşılamaktadır. Ancak yazılımı kullanarak neler yapıldığını/üretildiğini ve nasıl bir mekânda geçtiğini görünce "Dünya" çevirisinin anlamlı olduğu ortaya çıkmaktadır. Diğer taraftan haritadaki nesneleri programlarken verilen komutların yerleştirildiği paletlere "Kare" gibi bir çeviri verilmiştir. Yazılımda bu ve bunun gibi gerçek dünya ile tam olarak uyuşmayan örnekler tespit edilmiştir.

Sistem Durumunun Görünürlüğü başlığında (19 sorun bulunmuştur) ise yazılımların kullanımı esnasında o anki mevcut durum hakkında bilgilendirme, hatırlatma ve hata mesajlarının aynı yerde fark edilebilir bir şekilde gözükmemesi, nesne seçimini veya hareket ettirildiğini gösteren görsel bir bildirimin olmayışı, grafik arayüzlü menülerde hangi seçeneğin seçildiğinin belli olmaması gibi problemler tespit edilmiştir. Scratch yazılımı çok sık hata vermemekle birlikte oluşan durumun hata olup olmadığı da anlaşılamamaktadır. Hata mesajı alanı sabit bir

yerde çıkmayıp ekranın sol üst köşesinden başlayarak merdiven şeklinde her hata için bir alta inmektedir. Small Basic yazılımında ise menü kavramı bulunmadığı için herhangi bir seçim ve bu seçimi belli eden bir geribildirim bulunmamaktadır.

Yetenekler başlığında (18 sorun bulunmuştur), programlama öğretiminde kullanıcıların Small Basic ve Scratch yazılımlarının yetenekleri artırma ve uzmanlık kazandırma noktasında tam olarak istenileni karşılayamadığını gösteriyor.

Kullanıcı ile Seviyeli Bir İletişim başlığında (15 sorun bulunmuştur) Scratch ve Kodu Game yazılımlarının herhangi bir veri giriş alanında otomatik tamamlama (auto-complete) özelliğini sağlayamadığını söyleyebiliriz.

Estetik ve Sade Tasarım başlığında (10 sorun bulunmuştur) Small Basic yazılım incelendiğinde dışarıdan veri girişi yapabilmek için kutu dışında farklı veri girişi elemanları bulunmadığını görebiliyoruz. Aynı şekilde kod tabanlı bir arayüze sahip olan Robomind yazılımında ise istenen veri giriş nesneleri bulunmaktadır.

Esneklik ve Verimlilik başlığında (10 sorun bulunmuştur) Scratch ve Small Basic yazılımları incelendiğinde deneyimli ve deneyimsiz kullanıcılara göre hata oluştuğunda onların seviyesine göre hata mesajları oluşturmadığı görülmektedir. Özellikle Scratch bu bağlamda kimi durumlarda herhangi bir hata mesajı dahi vermemektedir. Kullanıcı kendi çabaları ve yeteneğine göre sorunu çözme yoluna gitmektedir.

Hatırlama Yerine Tanıma başlığında (8 sorun bulunmuştur) Small Basic yazılımı için hatırlama yerine tanıma veya anımsamayı sağlayacak anlamlı grupların kod tabanlı bir yazılım arayüzünde tam olarak sağlanamadığını görmekteyiz. Zira bu özellik menü alanında kısmen sunulsa da asıl işin yapıldığı kod alanında bulunmamaktadır.

b) Yazılımlar bazında oluşan sorunlara bakarak bir değerlendirme yapacak olursak;

Bünyesinde en çok sorunu (132) barındıran *Scratch* yazılımı, mevcut durum hakkında bilgilendirme ve hata mesajlarının aynı yerde çıkmaması, veri girişlerinin otomatik olarak tamamlanmaması, birçok işlem için geri al (*undo*) işlevinin olmayışı, az kullanılan görevlerin hatırlanamaması, sesli hata bildirimi eksikliği, hatanın nedeni ve ne yapılması gerektiği konusunda yönlendirilmemesi, gramer olarak hatalı mesajların çıkması, kullanıcıya kullanımdan kaynaklanan bir uzmanlık katmaması gibi konularda önemli eksiklikler tespit edilmiştir.

Small Basic yazılımında (127 sorun) ise hata yönetimi dediğimiz; hatanın sesle bildirimi, hatanın sorumlusunun sistem olduğunun vurgulanması, hata mesajlarında gramer olarak doğruluk, oluşan hatanın nedeni ve bundan sonra ne yapılması gerektiği gibi konuların ciddi eksiklikler olduğu farkedilmiştir. Bunun yanı sıra acemi ve uzman kullanıcılara karşı hata mesajlarının seviyelerinin uygun olmayışı, yine bu kullanıcılar için uygun bir yardım dokümantasyonun bulunmaması, uzun süreli kullanım sonrasında kullanıcıya yetenek ve uzmanlık kazandıracak kısayollar ve işlevlerden mahrum oluşu, veri girişi esnasında kolaylık sağlayacak radyo düğmeleri, açılır kutular gibi giriş nesnelerinin bulunmaması bir diğer önemli eksiklikler olarak karşımıza çıkmıştır.

Kodu Game yazılımında (115 sorun) ise, gerçek dünya ile uyuşmayan kültürel bir takım görsel unsurların oluşu, yine daha önce bahsettiğimiz hata yönetimi ve bildirimi problemleri, veri girişlerinin otomatik tamamlanmaması, dikey bir menünün olmayışı ve "çıkış" seçeneğinin bu menünün en altında bulunmaması gibi sorunlar bulunmuştur.

En az soruna (49) sahip *Robomind* yazılımında ise, nesne hareketi için girilen komutların sadece küçük harfle girilmesi sınırlaması, nadirde olsa meydana gelebilecek ve içinden

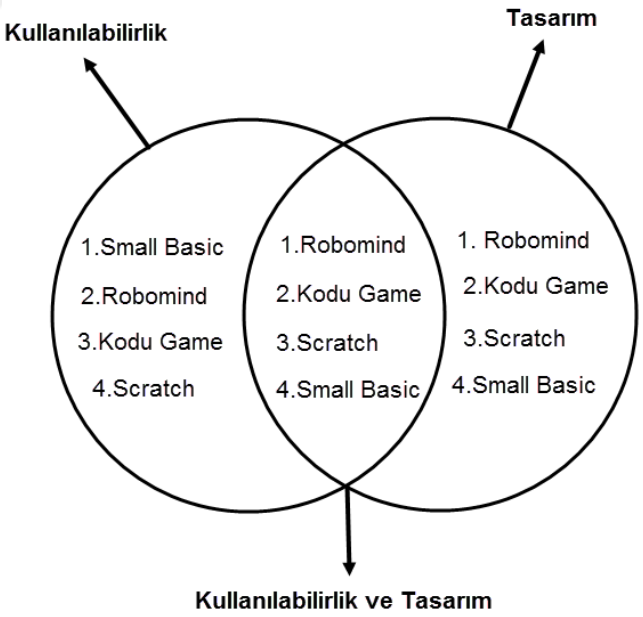
çıkılamayacak hata durumları, menülerin çoğunlukla dikey değil de yatay sunulması, oluşan hatalarda her iki kullanıcı tipi için mesaj seviyelerinin bulunmaması gibi problemler göze çarpmıştır.

c) Yazılımların barındırdığı yukarıdaki sorunlara kullanılabilirlik ve tasarım kavramları açısından bakılırsa;

Kullanılabilirlik ve tasarım kavramları bakımından Tablo 4.21'de yazılımların almış olduğu frekanslar gösterilmiştir. Bu verilere göre yazılımlarda bir sıralama yapacak olursa Şekil 5.1 elde edilecektir.

Şekil 5.1

Kavramlara Göre Görselleştirme Araçlarının Durumları



Şekil 5.1'e göre Small Basic yazılımı her iki kavramda en çok soruna sahip yazılım olarak bulunmuştur. Problem bakımından en çok sorunu olan Scratch yazılımı kullanılabilirlik ve tasarım kavramları bakımından üçüncü olmuştur. Kodu Game yazılımı ise bu alanda ikinci

olmuştur. Ancak BT öğretmenlerinin demografik bilgilerini sunduğumuz Tablo 4.4 ve Tablo 4.5'te Kodu Game yazılımının kullanılabilirlik ve tasarımının diğerlerinden (3 ve 4 puan toplamına göre) geride çıkmıştı. Bunun nedeni oyun üretim odaklı bir yazılım olduğu için oyun yaparken birçok kontrolün olması ve çok işlevsel bir arayüze sahip olması gösterilebilir. Bu da kullanıcılar tarafında aracın kullanılabilirlik ve tasarım algısını etkilemiş olabilir. Hem kullanılabilirlik hem de tasarım açısından istenen özelliklere sahip yazılım ise bu başlıkta birinci olarak Robomind yazılımı olmuştur.

Öneriler

- Small Basic yazılımı hata yönetimi, yardım dokümantasyonu, hata mesajları, kullanıcıya göre esnek kullanım gibi konularda ciddi iyileştirmelere ihtiyaç duymaktadır.
- Scratch yazılımı ise sık kullanılan işlevler için kısayollar, hatırlanabilme, hata yönetimi, kullanımda verimlilik gibi konularda düzeltmeler önerilmektedir.
- Microsoft Kodu Game yazılımının gerçek dünya ile simge uyumu, veri girişleri, menü tasarımı gibi görsel alana ilişkin problemleri çözülmelidir.
- Robomind yazılımında ise, kod girişinde küçük/büyük harf duyarlılığı, menü tasarımı, hata mesajlarında seviye durumları gibi konularda düzeltmeler önerilmektedir.
- Deniese Pierotti'nin 2005 yılında hazırlamış olduğu sezgisel kontrol aracı (Pierotti, 2015), gerek o yılların yazılım üretme altyapısı gerekse üreticilerin kullanıcı beklentilerine cevap vermesi bakımından mevcut teknolojik ortamın el verdiği ölçüde üretilen yazılımların tasarım ve kullanılabilirliğini değerlendirmede kullanılmıştır. Ancak gelişen teknoloji ve yazılım alt yapılarının farklı platformlara (web, telefon, tablet gibi) kaymasıyla ürün çeşitliliğinin artması neticesinde kullanılabilirlik sorunsalı çok daha farklı ve karışık bir boyuta taşınmıştır. İşletim sistemlerinin çoğalması, farklı internet tarayıcılarının ortaya çıkması, onlarca marka ve

modelde cep telefonu ve tablet cihazlarının doğuşuyla ve bunlarla birlikte gelen yazılımların tasarımlarının ne kadar uygun olduğu, kullanıcıya nasıl bir konfor sunduğunu tespit etmek güçleşmiştir. Bu bakımdan sezgisel kontrol aracının günümüz teknolojisinin sunduğu aygıtlara ve içerdiği yazılım/uygulamalara göre tekrardan düzenlenmesi gerekmektedir.

- Bu araştırma, yukarıda da bahsedildiği gibi mevcut ve geçerli bir sezgisel kontrol aracıyla 100 BT öğretmeninin görüşleri alınarak hazırlanmıştır. Fakat günümüz yazılım ortamları gereksinimlerine göre yeniden düzenlenmiş bir sezgisel kontrol aracıyla daha çok BT öğretmeni kullanılarak farklı araştırmalar yapılabilir.
- Her üretici firma programlama öğretimine farklı bir pencereden baktığı için üretilen yazılımların her bakımdan birbirlerine göre artı ve eksileri bulunabilmektedir. Öğreticiler bu gerçeği göz önüne alarak sınıf veya laboratuvarlarında programlama öğretimi sürecinde farklı yazılımları imkânlar el verdiği ölçüde kullanabilmelidir.
- Araştırma sonucunda sınıf ve laboratuvar ortamında kullanılabilirliği en uygun yazılım Robomind yazılımı çıkmıştır. Bu yazılım ilkokul ve ortaokul kademelerinde programlama öğretimi için gerek arayüz tasarımı gerekse işlevsellik yönünden ihtiyaçları karşılayabileceği düşünülmektedir. Robomind yazılımı diğerleri gibi bedava değil, ücretli bir yazılımdır. Öğrenci, öğretmen ve masaüstü şeklinde üç ayrı paket şeklinde satışı yapılmaktadır. Bu yönüyle ücretsiz olan diğer yazılımlara göre bir dezavantaja sahip olduğu söylenebilir. Ancak programlama öğretiminde yanlış ve eksik öğrenmeleri daha sonradan düzeltmenin getireceği maliyet düşünüldüğünde gerekli finansal desteğin sağlanmasıyla öğretimde kaliteli bir içerik ile ileriye dönük pozitif bir etki yaratacağı öngörülmektedir.
- Eğitimcilerin, programlama öğretiminde görselleştirme araçlarını kullanırken, konuya uygun araç seçiminin önemli olduğunu göz önünde bulundurmaları ve dersten önce uygun aracın

belirlenmesi ve derste hangi örnekler üzerinde durulacağı ile ilgili ön çalışma yapmaları gerekmektedir. Bravo, Marcelino, Gomes, Esteves ve Mendes (2005) bu konuda; algoritmayı bir karakter yardımıyla oluşturmaya yardımcı olan araçların, öğrencilerin programları anlamaları ve analiz etmeleri konusunda yardımcı olabileceğini belirtmişlerdir. Bu durumda ilköğretim öğrencilerine algoritma geliştirme ile ilgili eğitim verilirken görselliğin daha fazla ön planda olduğu bir yardımcı araç kullanımının öğrencilerin konuyu öğrenmelerini kolaylaştırdığı söylenebilir.

- Bir yazılımın çok farklı ve güzel özelliklerinin olması onun kullanılabilir olduğu anlamına gelmeyeceği için programlama öğretim sürecinde gerek arayüzdeki grafik unsurlarının yerleşimi ve sunuş tarzı gerekse de bunların işlevsel olması öğrenen ve öğreten açısından önemli hale gelmiştir. Programlama öğretimi sürecinde tasarım bakımından basit ve anlaşılır, kullanılabilirlik açısından kullanımı kolay ve esnek olan programlama öğretim yazılımları, ilkokul ve ortaokul kademelerinde yaşanabilecek programlama öğretim hatalarının önüne geçecektir. Böylelikle öğretim süreci daha kaliteli olup ve eğlenceli bir hal alacak, yanlış öğrenmelere meydan vermeyerek zaman kaybı yaşanmayacaktır.

Son Söz

Sonuç olarak, modern dünyada bilgiye dayalı yeni bir ekonomi kurulmaktadır. Bu ekonomide rekabet edebilmek için ileri teknolojiye dayalı sektörlerde çalışacak insanları yetiştirmemiz gerekmektedir. Altın, elmas, petrol, doğalgaz gibi kaynakları bulunmayan bir ülke olarak dünya ile rekabet halinde olabilmemiz için mutlaka yeni nesillere erken yaşlarda bilgisayar programlamayı öğretmemiz şarttır. Bu manada okul, sınıf ve laboratuvarlarımızda soyut ve karışık zihinsel işlemler süreci gerektiren bir programlama öğretimi yerine öğrencilerimizin yaş ve seviyelerini dikkate alarak tasarım ve kullanılabilirlik şartlarını taşıyan, algoritma mantığını

kavratan, programlamayı basit anlamda ele alan ve sevdiren görselleştirme araçlarını kullanmaya ihtiyacımız vardır. Pek çok ülkede çoktandır yapılan bu tür bir programlama öğretimi sayesinde teknolojiye dayalı sektörlerde bilişimle üretim yapabilir ve ülke olarak kalkınabiliriz.

Kaynakça

Akçay, T. (2009). *Perceptions of students and teachers about the use of a kid's programming language in computer courses.* (Unpublished master's thesis). Middle East Technical University, Ankara.

Alice3D. (2015). *Alice 3D eğitim*. http://alice3degitim.weebly.com 'den alınmıştır.

Arabacıoğlu, T., Bülbül, H. İ.,& Filiz, A. (2007). Bilgisayar programlama öğretiminde yeni bir yaklaşım. IX. Akademik Bilişim Konferansı bildirileri içinde Dumlupınar Üniversitesi, Kütahya: Nokta Matbaacılık. http://ab.org.tr/ab07/kitap/arabacioglu_bulbul_AB07.pdf 'den alınmıştır.

Aşkar, P. (1990). *Okullarda bilgisayar destekli öğretim uygulamaları.* Ankara: BITAV.

Baldwin, L. P.,& Kuljis, J. (2001). Learning programming using program visualization techniques. Proceedings of the 34th Hawaii International Conference içinde Maui, HI, USA: IEEE. https://www.computer.org/csdl/proceedings/hicss/2001/0981/01/09811051.pdf 'den alınmıştır.

Barnum, C. M.,& Dragga, S. (2001). *Usability testing and research.* USA: Longman Publishing Group.

Bayman, P.,& Mayer, R. (1988). Using conceptual models to teach BASIC computer programming. *Journal of Educational Psychology, 80*(3), 291-298.

Bergin, J.,& Martinez, M. P. (1996). An overview of visualization: Its use and design: Report of the working group in visualization. Integrating Technology into Computer Science

Education 6/96 içinde (ss. 192-200). New York: ACM. http://lsd.ls.fi.upm.es/lsd/papers/1996/iticse96-wg.pdf 'den alınmıştır.

Bevan, N. (2000). *ISO and industry standards for user centred design.* UK: Serco Ltd.

Bilgiustam. (2015). *Programlama dili nedir? Türleri nelerdir?* http://www.bilgiustam.com/programlama-dili-nedir-turleri-nelerdir 'den alınmıştır.

Bishop-Clark, C., Courte, J.,& Howard, E. (2007). A quantitative and qualitative investigation of using ALICE programming to improve confidence, enjoyment and achievement among non-majors. *Journal of Educational Computing Research, 37*(2), 193-207.

Boren, T.,& Ramey, J. (2000). Thinking aloud: Reconciling theory and practice. *IEEE Transactions on Professional Communication, 43*(3), 261-278.

Bozkurt, M. (2015). *Nicel ve nitel araştırma yöntemleri.* https://prezi.com/r0xpbqkfmhxc/nicel-ve-nitel-arastrma-yontemleri 'den alınmıştır.

Bravo, C., Marcelino, M. J., Gomes, A., Esteves, M.,& Mendes, A. J. (2005). Integrating educational tools for collaborative computer programming learning. *Journal of Universal Computer Science, 11*(9), 1505-1517.

Brinck, T., Gergle, D.,& Wood, S. D. (2001). *Designing web sites that work: usability for the web.* San Francisco: Morgan Kaufmann Publishers Inc.

Brooke, J. (1996). SUS: *A "quick and dirty" usability scale. Usability evaluation in industry*. http://www.usabilitynet.org/trump/documents/Suschapt.doc 'den alınmıştır.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A.,& Miller, P. (1997). Mini-languages: A way to learn programming principles. *Education and Information Technologies, 2*(1), 65-83.

Bush, V. (1945). As we may think. *The Atlantic.* http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881 'den alınmıştır.

Buurman, R. (1997). User centered design of smart objects. *Ergonomics, 40*(10), 1159-1169.

Bülbül, H. İ. (1999). Öğretim amaçlı bilgisayar yazılımlarında ekran tasarımı. *Milli Eğitim Dergisi*. 141, Ankara,Türkiye. http://dhgm.meb.gov.tr/yayimlar/dergiler/Milli_Egitim_Dergisi/144/bulbul.htm 'den alınmıştır.

Calder, N. (2010). Using Scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom, 15*(4), 9-14.

Card, S., Moran, T.,& Newell, A. (1983). *The psychology of human computer interaction.*New Jersey: Lawrence Erlbaum Associates.

Casey, P. (1997). Computer programming. *Journal of Computers in the Schools, 13*(1-2), 41-51.

Chen, T.,& Sobh, T. (2001). A tool for data structure visualization and user-defined algorithm animation. 31st ASEE IEEE Frontiers in Education Conference içinde (ss. 2-7). Reno, NV: IEEE. www1bpt.bridgeport.edu/~risc/pdf/jp29.pdf 'den alınmıştır.

Chisman, J., Diller, K.,& Walbridge, S. (1999). Usability testing: A case study. *College & Research Libraries,* (60), 552-569.

Choi, H. (2012). Learners' reflections on computer programming using Scratch: Korean primary pre-service teachers' perspective. 10th International Conference for Media in Education içinde, Beijing Normal University, China.

Choi, S. K., Bell, T., Jun, S. J.,& Lee, W. G. (2008). Designing offline computer science activities for the Korean elementary school curriculum. Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education içinde (ss. 338-338). New York, USA: ACM. http://www.cosc.canterbury.ac.nz/tim.bell/cseducation/papers/Choi%20Bell%20Jun%20Lee%202008%20ITiCSE%20full%20version.pdf 'den alınmıştır.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programmimg on young children's cognition. *Journal of Educational Psychology, 6*(76), 1051-1058.

Colin, A., Gray, G., Pavlos, S., & Athanaios, T. (2005). Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing (JERIC), 5*(3). doi:10.1145/1163405.1163410.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching object-first in introductory computer science. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education içinde (ss. 191-195). New Yok: ACM. http://www.alice.org/publications/pubs/TeachingObjectsfirstInIntroductoryComputerScience.pdf 'den alınmıştır.

Cooper, S., Powers, K., McNally, M., Goldman, K., Proulx, V., & Carlisle, M. (2006). Tools for teaching introductory programming:What works? 37th SIGCSE Technical Symposium on

Computer Science Education içinde (ss. 560-561). Housten, Texas, USA. http://dsys.cse.wustl.edu/resources/papers/gross-sigcse-2006.pdf 'den alınmıştır.

Corry, M., Frick, T., & Hansen, L. (1997). User-centered design and usability testing of a web site: An illustrative case study. *ETR & D, 45*(4), 65-76.

Crews, T., & Ziegler, U. (1998). The flowchart interpreter for introductory programming course. Frontiers in Education Conference içinde (ss. 307-312). Tempe, Arizona: IEEE.

Çağıltay, K. (2011). *İnsan bilgisayar etkileşimi ve kullanılabilirlik mühendisliği: Teoriden pratiğe.* Ankara: ODTÜ Yayıncılık.

Çetin, E. (2012). *Bilgisayar programlama eğitiminin çocukların problem çözme becerilerine etkisi.* (Yüksek Lisans Tezi). Gazi Üniversitesi, Eğitim Bilimleri Enstitüsü, Ankara. (Erişim No. 349116).

Çetin, H. (2002). Liberalizmin tarihsel kökenleri. *Cumhuriyet Üniversitesi İktisadi ve İdari Bilimler Dergisi, 3*(1), 79-96.

Çölkesen, R. (2002). *Veri yapıları ve algoritmalar.* İstanbul: Papatya Yayıncılık.

Dann, W., Cooper, S., & Pausch, R. (2000). Alice : A 3d tool for introductory programming concepts. *The Journal of Computing in Small Colleges, 15*(5), 107-116.

Davis, F. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 19-337.

Dershem, H. L., & Brummund, P. (1998). Tools for web-based sorting animation. Proceeding of the 29th Annual SIGCSE Conference on Technical Symposium on Computer Science Education içinde (ss. 222-226). New York: ACM.

Dickstein, R., & Mills, V. (2000). Usability testing at the university of Arizona library: How to let the users in on the design. *Information Technology & Libraries, 19*(3), 144-151.

Drucker, P. (2000). *Yeni gerçekler.* Ankara: Türkiye İş Bankası Yayınları.

Dumas, J. (2007). The great leap forward: The birth of the usability profession. *Journal of Usability Studies, 2*(2), 54-60.

Dumas, J., & Redish, J. (1993). *A practical guide to usability testing.* Norwood, NJ: Ablex Publisihing Group Co.

EBA. (2015). *Microsoft KODU Game Lab.* http://www.eba.gov.tr/haber/1382225534 'den alınmıştır.

Erdoğan, B. (2005). *Programlama başarısı ile akademik başarı, genel yetenek, bilgisayara karşı tutum, cinsiyet ve lise türü arasındaki ilişkilerin incelenmesi.* (Yüksek Lisans Tezi). Marmara Üniversitesi, Eğitim Bilimleri Enstitüsü, İstanbul. (Erişim No. 188754).

Ericson, K., & Simon, H. (1993). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Ersoy, H., Madran, R. O., & Gülbahar, Y. (2011). Programlama dilleri öğretimine bir model önerisi: Robot programlama. XIII. Akademik Bilişim Konferansı bildirileri içinde Malatya: İnönü Üniversitesi. http://ab.org.tr/ab11/bildiri/145.pdf 'den alınmıştır.

Eryılmaz, S. (2003). *Algoritma tasarlama ve programlamaya giriş.* Ankara: Detay Yayıncılık.

Esteves, M., & Mendes, A. J. (2004). A simulation tool to help learning of object oriented programming basics. Frontiers in Education Conference içinde (ss. 7-12). Savannah, GA.

Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with Scratch on future teachers' opinions and attitudes about programming and ICT in education. *ACM SIGCSE, 41*(3), 258-262.

Garner, S. (2003). Learning resources and tools to aid novices learn programming. Informing Science ve Information Technology Education Joint Conference içinde (ss. 213-222). Pori, Finland. http://www.proceedings.informingscience.org/IS2003Proceedings/docs/036Garne.pdf 'den alınmıştır.

Gilmore, D. (1990). Methodological issues in the study of programming. In J. Hoc, T. Green, & R. Samurcay (Eds.), *Psychology of programming* (pp. 83-98). London: Academic Press.

Goldenson, D. (1996). Why teach computer programming? Some evidence about generalization and transfer. National Educational Computing Conference içinde Minneapolis: MN. ftp://ftp.sei.cmu.edu/pub/emg/transferPaper/necc'96.pdf 'den alınmıştır.

Goodwin, N. (1987). Functionality and usability. *Communications of the ACM, 30*(3), 229-233.

Gülmez, I. (2009). *Programlama öğretiminde görselleştirme araçlarının kullanımının öğrenci başarı ve motivasyonuna etkisi.* (Yüksek Lisans Tezi). Marmara Üniversitesi, İstanbul. (Erişim No. 250819).

Gültekin, K. (2006). *Çokluortamın bilgisayar programlama başarısı üzerine etkisi.*(Yüksek Lisans Tezi). Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara. (Erişim No. 182319).

Hartson, H. R. (1998). Human-computer interaction: Interdisciplinary roots and trends. *The Journal of Systems and Software, 43*(2), 103-118.

Head, A. (1999). Web redemption and the promise of usability. *Online*, (23), 20-28.

Hewett, T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., &Verplank, W. (2004). Curricula for human-computer interaction. *ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group*.

Horie, S., Isomura, A., Murase, Y., Ohwada,T., Yatabe, Y., Kishiue, K., & İrikata, S. (2008). Usability Evaluation in Product Development. *Mitsubishi Motors Technical Review, 20,* 113-117.

Horn, J. (2015). *The usability methods toolbox handbook*. https://www.yumpu.com/en/document/view/11598336/the-usability-methods-toolbox-handbook-james-hom 'den alınmıştır.

Hu, M. (2004). Teaching novices programming with core language and dynamic visualization. *17th National Advisory Committee on Computing Qualifications* (pp. 94-103). Christchurch, New Zealand.

Hundhausen, C., Douglas, J., & Stasko, S. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing, 13*(3), 259-290.

Hutchinson, H., Druin, A., & Bederson, B. (2007). Designing Searching and Browsing Software for Elementary-Age Children.In J. Lazar (Ed.), *Universal usability: Designing computer interfaces for diverse users.* (pp.13-42). USA: John Wiley and Sons.

Hyrskykari, A. (1993). Development of program visualization systems. *2nd Czech British Symposium of Visual Aspects of Man-Machine Systems* Praha.

IBM. (2015). *IBM design studio*. https://www.ibm.com/design/studio.shtml 'den alınmıştır.

ISO. (1994). Ergonomic requirements for office work with visual display terminals (VDTs) Part 11: Guidance on usability. *(ISO DIS 9241-11)*. London: International Standards Organization.

İnce, İ., Şenyüzlü, B., & Uğur, B. (2007). *İlköğretim bilişim teknolojileri 6, 7 ve 8. basamak öğretmen kılavuz kitabı.* Ankara: MEB.

İpek, İ. (2001). *Bilgisayarla öğretim tasarım, geliştirme ve yöntemler.* Ankara: Tıp Teknik Yayınevi.

Jeng, J. (2006). *Usability of the digital library: An evaluation model.*(Unpublished doctoral dissertation). The State University of New Jersey, New Brunswick.

Jenkins, T. (2002). On the difficulty of learning to program. 3rd annual Conference of LTSN-ICS içinde University of Leeds, Leeds, UK. http://www.psy.gla.ac.uk/~steve/localed/jenkins.html 'den alınmıştır.

Jenkins, T., & Davy, J. (2002). Diversity and motivation in introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences, 1*(1). 1-9.

Kahn, K. (1996). ToonTalk-an animated programming environment for children. *Journal of Visual Languages & Computing*, *7*(2), 197-217.

Kalyuga, S., Chandler, P., & J.Sweller. (2001). Learner experience and efficiency of instructional guidance. *Educational Psychology*, *21*(1), 5-23.

Karasar, N. (2013). *Bilimsel araştırma yöntemi.* Ankara: Nobel Yayıncılık.

Kaucic, B., & Asic, T. (2011). Improving introductory programming with Scratch. Proceedings of the 34th International Convention içinde (ss. 1095 - 1100). Opatija: IEEE

Kazu, İ., & Yavuzalp, N. (2008). Öğretim yazılımlarının kullanımına ilişkin öğretmen görüşleri. *Eğitim ve Bilim, 33*(150). 110-126.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. SIGCHI Conference on Human Factors in Computing Systems Table of Contents içinde (ss. 1455-1464). San Jose, California, USA. http://www.cs.wustl.edu/~ckelleher/StorytellingCHI.pdf 'den alınmıştır.

Kesseler, E., & Knapen, E. (2006). Towards human centered design, two case studies. *Journal of Systems and Software, 79*(3), 301-313.

Klassen, M. (2006). Visual approach for teaching programming concepts. 9th International Conference on Engineering Education içinde (ss. 141-145). New York: ACM. http://public.clunet.edu//~mklassen/ICEE2006.pdf 'den alınmıştır.

Korhonen, A. (2003). *Visual algorithm simulation.* (Doctoral dissertation). Helsinki University, Technology Department of Computer Science and Engineering, Espoo, Finland. http://www.cs.hut.fi/Research/SVG/publications/tkoa40.pdf 'den alınmıştır.

Küçükkalay, M. (1997). Endüstri devrimi ve ekonomik sonuçlarının analizleri. *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 2*(2), 57-60.

Lahtinen, E., Ahoniemi, T., & Salo, A. (2007). Effectiveness of integrating program visualizations to a programming course. 7th Baltic Sea Conference on Computing Education Research içinde (ss. 195-198). Koli, Finland. http://crpit.com/confpapers/CRPITV88Lahtinen.pdf 'den alınmıştır.

Lai, S., & Repman, J. L. (1996). The effects of analogies and mathematics ability on students' programming learning using computer-based learning. *International Journal of Instructional Media, 23*(4), 355-364.

Landauer, T. (1995). *The trouble with computers: Usefulness, usability and productivity.* Cambridge, MA: MIT Press.

Lewis, J. (1991). Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the ASQ. *ACM SIGCHI Bulletin, 23*(1), 78-81.

Lewis, J. (1995). IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction, 7*(1), 57-78.

Lin, C., & Zhang, M. (2003). The use of computer animation in teaching discrete structures course. The 36th Annual Midwest Instruction and Computing Symposium içinde MICS. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.7565&rep=rep1&type=pdf

Lin, J. M., Yen, L. Y., Yang, M., & Chen, C. (2005). Teaching computer programming in elementary schools: A pilot study. National Educational Computing Conference Center içinde Philadelphia, PA. http://www.stagecast.com/pdf/research/Lin_NECC2005_Paper_RP.pdf 'den alınmıştır.

Maguire, M. (2001). Methods to support human-centered design. *International Journal, 55*(4), 587-634.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. 38th ACM Technical Symposium on Computer Science Education içinde (ss. 223-227). Covington, Kentucky: ACM.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4). 16.

Mandir, S. S. (2007). *Model curriculum and teaching material for K-12 Indian schools.* http://www.it.iitb.ac.in/~sri/papers/SSRVM-CS-March07.pdf 'den alınmıştır.

McCormic, E., & Sanders, M. (1993). Human Factors In Engineering and Design. *Human factors in system design.* New York, USA: McGraw-Hill.

McGill, T. J., & Volet, S. E. (1997). A conceptiual framework for analyzing students' knowledge of programming. *Journal of Research on Computing in Education, 29*(3), 276-297.

Medyanadolu. (2015). *Yazılım geliştirme sürecinde tasarım aşaması*. http://www.medyanadolu.com/blog/index.php/web-tasarim/web-sitesi-uretimi-ve-proje-yonetim-sureci.html 'den alınmıştır.

Meisalo, V., Suhonen, J., Torvinen, S., & Sutinen, E. (2002). Formative evaluation scheme for a web-based course design. 7th Annual Conference on Innovation and Technology in Computer Science Education içinde (ss. 130-134). New York, NY, USA: ACM.

Merill, D. (1992). *Computers in education.* U.S.A: Allyn and Pacon A Division of Simon ve Schuster Inc.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on out capacity for processing information. *Psychological Review, 63*(2), 81-97.

Miyadera, Y., Huang, N., & Yokoyama, S. (2000). A programming language education system based on program animation. IFIP 16th World Computer Congress içinde (ss. 258-261).

Moore, C. (1997). *Product usability.* http://www.sevenwoodsaudio.com/AN7.pdf 'den alınmıştır.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. International Working Conference on Advanced Visual Interfaces (ss. 373-376). Gallipoli, Italy: AVI. http://cs.joensuu.fi/jeliot/files/avi04.pdf 'den alınmıştır.

Naps, T. (2005). JHave – adressing the need to support algorithm visualization with tools for active engegement. *IEEE Computer Graphics and Applications, 25*(5). 49-55.

Naps, T. L., Eagan, J., & Norton, L. L. (2000). jHave- an environment to actively engage students in web-based algorithm visualisations. Proceeding of the 31th SIGCSE Technical Symposium on Computer Science Education içinde (ss. 109-113). New York, NY: ACM SIGCSE Bulletin. https://perso.telecom-paristech.fr/~eagan/media/papers/naps00.pdf 'den alınmıştır.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C.,& Rodger, S. (2003). Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin, 35*(2), 131-152.

Nelson, M., & Rice, D. (200). Introduction to algorithms and problem solving. 30th ASEE/IEEE Frontiers in Education Conference içinde (ss. S2C-5). Kansas City, MO: IEEE.

Nielsen, J. (1994a). *Usability engineering.* Boston, MA: AP Professional.

Nielsen, J. (1994b). *Usability inspection methods.* New York: John Willey&Sons.

Nielsen, J. (2015a). *10 usability heuristics for user interface design*. http://www.nngroup.com/articles/ten-usability-heuristics 'den alınmıştır.

Nielsen, J. (2015b). *Why you only need to test with 5 users*. http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users 'den alınmıştır.

Norman, D. (1988). *Design for everyday things.* New York: Doubleday.

Olson, G. M., & Olson, J. S. (2003). Human-computer interaction: Psychological aspects of the human use of computing. *Annual Review of Psychology, 54*, 491-516.

Oxford (2015). In *Oxford Dictionaries.* http://www.oxforddictionaries.com/definition/english/usable?q=usability#usable__6 'den alınmıştır.

Özcan, O. (2008). *İnteraktif medya tasarımında temel adımlar.* İstanbul: Pusula Yayıncılık.

Pelgrum, W., & Plump, T. (1991). *The use of computer in education worldwide.* Pergamon Press Inc.

Peppler, K., & Kafai, Y. (2007). What video game making can teach us about learning and literacy: Alternative pathways into participatory cultures. Proceeding of the Digital Games Research Association Conference içinde (ss. 369-376). Tokyo, Japan. http://www.digra.org/dl/db/07311.33576.pdf 'den alınmıştır.

Perry, G. (2009). *Yeni başlayanlar için programlama kılavuzu.* (T. Aksoy, Çev.) İstanbul: Sistem Yayıncılık.

Pheasant, S. (2003). *Bodyspace antropometry, ergonomics and the design of work* (2nd ed.). London: Taylor & Francis Ltd New Fetter Lane.

Pierotti, D. (2015). *Heuristic evaluation – A system checklist.* https://web.fe.up.pt/~ei08119/wiki/lib/exe/fetch.php?media=heuristic_evaluation_-_system_checklist.pdf 'den alınmıştır.

Prawalpatagool, J. (2010). *An effective technique for learning in the computer programming subject.* (Master's thesis). Management of Information Technology, Prince of Songkla University.

Preece, J. (Ed.). (1994). *Human-computer interaction.* USA: Addison Wesley.

Programlar. (2015). *KODU Game Lab*. http://www.programlar.com/makale/microsoft-bir-sonraki-neslin-oyun-yapimcilarini-yetistiriyor-kodu-game-lab.3a.5.1.html 'den alınmıştır.

Proulx, V. (2000). Programming patterns and design patterns in the introductory computer science course. *SIGCSE Bulletin, 32*(1), 80-84. http://www.ccs.neu.edu/home/vkp/Papers/Patterns-sigcse2000.doc 'den alınmıştır.

Raeder, G. (1985). A survey of current graphical programming techniques. *IEEE Computer*, *18*(8). 11-25.

Rajala, T., Laakso, M., Kaila, E., & Salakoski, T. (2008). Effectiveness of program visualization:A case study with the ViLLE tool. *Journal of Information Technology Education: Innovations in Practice, 7*, 15-32.

Ramadan, H. (2000). Programming by discovery. *Journal of Computer Assistes Learning, 16*(1), 83-93.

Redish, J. (2007). Expanding usability testing to evaulate complex system. *Journal of Usability Studies, 2*(3), 102-111.

Resnick, M. (2007). All i really need to know (about creative thinking) i learned (by studying how children learn) in kindergarten. Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition içinde (ss. 1-6). New York: ACM. http://web.media.mit.edu/~mres/papers/kindergarten-learning-approach.pdf 'den alınmıştır.

Rızvanoğlu, K. (2009). *Herkes için web: Evrensel kullanılabilirlik ve tasarım.* İstanbul: Punto Yayınları.

Robomind. (2015). *Robomind.net*. http://robomind.net/tr/index.html 'den alınmıştır.

Rodger, S. (2002). *Using hands-on visualisation to teach computer science from beginning courses to advanced courses*. http://www.cs.duke.edu/~rodger 'den alınmıştır.

Rubin, J. (1994). *The handbook of usability testing: How to plan, design, and conduct effective tests.* New York: John Wiley.

Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing , How to Plan Design, and Conduct Effective Test.* (2nd ed.). Indianapolis, Indiana: Wiley Publishing.

Saeli, M., Perrenet, J., Jochems, W. M. & Zwaneveld, B. (2011). Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics and Education, 10*(1), 73-88.

Schackel, B. (1997). Human-computer interaction - whence and whither? *Journal of the American Society for Information Science, 48*(11), 970-986.

Schifferstein, H. N., & Hekkert, P. (2008). *Product experience*. Hungary: Elsevier.

ScratchÖgren. (2015). *Scratch nedir?* http://scratchogren.com/scratch-nedir 'den alınmıştır.

Shneiderman, B. (1998). Human Factors of Interactive Software. *Designing the user interface:Strategies for effective human-computer interaction.* Addison Wesley.

Sleeman, D., Putnam, R. T., Baxter, J., & Kuspa, L. (1984). Pascal and high-school students: a study of misconceptions. *Technology Panel Study of Stanford and the Schools* (pp. 34). Stanford Univ, CA.

Smith, D. C., Cypher, A., & Tesler, L. (2000). Novice programming comes of age. *Communications of the ACM, 43*(3), 75-81.

Smith, R. S. (2015). *Interface design for educational multimedia*. http://ihashimi.aurasolution.com/multimedia/Interface_Design_Educational_Media.pdf 'den alınmıştır.

Stephenson, C. (2001). Knowing what to teach-computing in high schools. The Computer Science and Information Technology Symposium Issues and Trends in High School Computing içinde (ss. 15). Chicago, Illinois.

Suchman, L. (Ed.). (1987). *Interactive artifacts:Plans and situated actions.* New York: Cambridge University Press.

Şengel, E., & Özdemir, S. (Editörler). (2012). *Web siteleri için kullanılabilirlik ölçümleri.* Bursa: Ekin Basın Yayın Dağıtım.

Şeniş, F. (1991). Bilgisayar destekli öğretim yazılımlarında standart sorunu. Eğitim Teknolojisi ve Bilgisayar Destekli Eğitim 1. Sempozyumu bildirileri içinde (ss. 183-191). Eskişehir: Anadolu Üniversitesi.

TDK. (2015). *Türk Dil Kurumu.* Ankara: TDK.

Telek, C. (2013). *Kullanılabilirlik kavramı,tasarım süreci içindeki yeri ve benzer tasarım yaklaşımları ile ilişkisi.* (Yüksek Lisans Tezi). Mimar Sinan Güzel Sanatlar Üniversitesi Fen Bilimleri Enstitüsü, İstanbul. (Erişim No. 328505).

Thomas, R. (1998). *Elements of performance and satisfaction as indicators of the usability of digital spatial interfaces for information seeking: Implications for ISLA.* (Unpublished doctoral dissertation). Graduate School of Southern California.

Traynor, D., & Gibson, P. (2004). Towards the development of a cognitive model of programming; A software engineering approach. 16th Meeting Of The Psychology Of Programming Interest Group içinde Carlow, Ireland. http://www.cs.may.ie/~dtraynor/papers/PPIGarticle.pdf 'den alınmıştır.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003). A model curriculum for k12 computer science. *Final Report of the Association for Computing Machinery* New York: ACM. http://dl.acm.org/citation.cfm?id=2593247 'den alınmıştır.

Üstündağ, Ö. (1999). *Bilgisayar arayüz tasarımında grafik elemanlar ve İnternet'in grafiksel etkileşim arayüzü: World Wide Web.* (Yüksek Lisans Tezi). Hacettepe Üniversitesi Sosyal Bilimler Enstitüsü, Ankara. (Erişim No. 81982)

Vikipedi. (2015a). *Programlama*. http://tr.wikipedia.org/wiki/Programlama 'den alınmıştır.

Vikipedi. (2015b). *Programlama dili*. http://tr.wikipedia.org/wiki/Programlama_dili 'den alınmıştır.

Vikipedi. (2015c). *Yorumlayıcı*. https://tr.wikipedia.org/wiki/Yorumlayıcı 'den alınmıştır.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). The cognitive walkthrough method: A practitioners guide. In J. Nielsen, & R. L. Mack (Eds.), *Usability inspection methods* (pp.105-140). New York: Wiley.

Wickens, C. (1992). *Engineering psychology and human performance.* New York: Harper Collins.

Wickens, C. D., J. Lee, Y. L., & Becker, S. G. (2004). *An introduction to human factors engineering* (2nd ed.). New Jersey: Pearson Education International.

Yalın, H. (2001). *Öğretim yeknolojileri ve materyal geliştirme.* Ankara: Nobel Yayın Dağıtım.

Ekler

Ek 1- Sezgisel Kontrol Aracı

Programlama Öğretimi Araçlarının Kullanılabilirlik ve Tasarım Anketi

Değerli meslektaşım;
Bu anket Uludağ Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Eğitimi Tezli Yüksek Lisans çalışması kapsamında yapılmaktadır. Sizlerden, size gönderilen e-postada kullanmanız istenen programlama öğretimi aracına göre bu anketi doldurmanız beklenmektedir. Ankette bu araçlar; Kullanılabilirlik ve Tasarım olarak iki ana kategoriye göre 12 alt başlıkta değerlendirilmektedir. Amacımız programlama öğretiminde kullanılan bu araçların, ne kadar kullanılabilir olduğu ve tasarımındaki eksiklikleri tespit etmektir. Bu bağlamda çalışmanın programlama öğretimi metodolojisine, Bilişim Teknolojileri ve Yazılım müfredatına ve bize katkısının olacağını düşünüyorum.

Anketi doldururken göstereceğiniz hassasiyet ve sabırdan dolayı şimdiden teşekkür ederim.

Salih BALTALI
Bilişim Teknolojileri Öğretmeni
Bursa/Nilüfer Zekai Gümüşdiş Ortaokulu

* Gerekli

Demografik Bilgiler

Size ve deneyimlerinize dair bilgilerdir. Hiçbir surette paylaşım olmayacaktır.

1-Cinsiyetiniz *

2-Görev yaptığınız ilçe neresidir? *

3-Hizmet yılınız nedir? *

4-Hangi programlama öğretimi aracını kullandınız? *
Burada size gönderilen e-postada sizin kullanmanızı istediğimiz programlama öğretimi aracı kastedilmektedir.

5-Programlama bilginizi derecelendirseydiniz kendinize kaç verirdiniz? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

6-Programlama öğretiminde görsel araçların kullanılmasını nasıl değerlendiriyorsunuz? *

- ○ Yararlı olacağını düşünmüyorum
- ○ Kararsızım.
- ○ Çok faydasının dokunacağı kanaatindeyim.

7-Kullandığınız programlama öğretimi aracının hangi yaş grubuna hitap ettiğini düşünüyorsunuz? *

8-Sizce bu tür araçlarda Kullanılabilirlik veya Tasarımdan hangisine daha çok dikkat edilmelidir? *

- ○ Kullanılabilirlik
- ○ Tasarım
- ○ Her ikiside
- ○ Hiçbiri

9-Kullandığınız aracın Kullanılabilirliğini nasıl derecelendirirsiniz? *

	1	2	3	4	5	
Çok kötü	○	○	○	○	○	Çok iyi

10-Kullandığınız aracın Tasarımını nasıl derecelendirirsiniz? *

	1	2	3	4	5	
Çok kötü	○	○	○	○	○	Çok iyi

1.Sistem Durumunun Görünürlüğü

Sistem, kullanıcıyı sürekli olarak o anki durumunun nasıl gittiği konusunda sürekli bir şekilde uygun dönütler eşliğinde bilgilendirmelidir.

1-Tüm sistemde gösterge (ikon) tasarım ve stil şekilleri tutarlı mı? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

2-Her menüdeki bilgilendirmeler, hatırlatmalar ve hata mesajları sistem içerisinde aynı yerde görünmekte midir? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

3-Eğer hata mesajları açılan bir alan içerisinde gösteriliyor ise, kullanıcı, hatalı olan alanı görebilmekte midir? *

- Evet
- Hayır
- Uygulaması Yok

4-Nesnelerin seçildiğini veya hareket ettirildiğini gösteren görsel geribildirim mevcut mudur? *

- Evet
- Hayır
- Uygulaması Yok

5-Grafik arayüzlü menülerde, hangi seçeneğin seçildiği açık şekilde belli oluyor mu? *

- Evet
- Hayır
- Uygulaması Yok

2. Sistem ve Gerçek Dünyanın Eşleşmesi

Sistem içerisinde kullanılan kelimeler, kavramlar, cümleler kullanıcıya tanıdık olmalıdır ve teknik bir dil kullanılmasından kaçınılmalıdır. Bilgi mantıksal sırasında ve doğallığında gösterilerek gerçek dünyada olan düzeninde sunmalıdır.

1-Sistem içerisinde yer alan göstergeler (ikon), kullanıcı için somut ve tanıdık mıdır? *

- Evet
- Hayır
- Uygulaması Yok

2-Eğer görsel bir ipucu olarak kullanılan şekil varsa, bu kültürel geleneklerle eşleşiyor mu? *

- Evet
- Hayır
- Uygulaması Yok

3-Seçilen renkler, renk kodları ile ilgili genel beklentileri karşılıyor mu? *

- Evet
- Hayır
- Uygulaması Yok

4-Hatırlatmalar, sistem içerisinde gerekli hareketleri açıklarken, kullanılan mesaj açıklanan harekete uygun mudur? *

- Evet
- Hayır
- Uygulaması Yok

5-Menü başlıkları dilbilgisi yönünden birbirleri ile tutarlı mıdır? *

- Evet
- Hayır
- Uygulaması Yok

6-Sistem kullanıcı jargonunu kullanıp ve bilgisayar jargonundan sakınmakta mıdır? *

- Evet
- Hayır
- Uygulaması Yok

7-Komut isimleri anlamlı mıdır? *

- Evet
- Hayır
- Uygulaması Yok

3. Kullanıcı Kontrol ve Özgürlüğü

Kullanıcılar sık sık sistem fonksiyonlarının seçiminde hata yapar ve bu istenmeye durumda çok detaya girmeden çıkmak için açıkça belirtilmiş bir "acil çıkış"a ihtiyaç duyarlar. Geri alma (undo) ve yeniden yapma (redo) seçenekleri bu amaçla sunulmaktadır.

1-Seyrek (az) olarak kullanılan görevlerin hatırlanması kolay mıdır? *

- Evet
- Hayır
- Uygulaması Yok

2-Açılır pencere olan sistemlerde, kullanıcılar için pencereler arası geçiş kolay mıdır? *

- Evet
- Hayır
- Uygulaması Yok

3-Herhangi bir işlem hareketi, veri girişi veya birçok diğer işlem hareketleri için "geri al" fonksiyonu bulunmakta mıdır? *

- Evet
- Hayır
- Uygulaması Yok

4-Kullanıcılar, devam eden işlemleri iptal edebiliyorlar mı? *

- Evet
- Hayır
- Uygulaması Yok

5-Sistem, kopyalayarak veya varolan veriyi değiştirerek kullanıcının veri giriş süresini azaltıyor mu? *

- Evet
- Hayır
- Uygulaması Yok

6-Kullanıcıların menü öğelerini tıklayarak veya klavye kısayolunu kullanarak seçme olanağı var mı? *

- Evet
- Hayır
- Uygulaması Yok

7-Menüler iç içe olmaktan ziyade daha geniş (bir menüde birden fazla öğe,ribon tarzı) mi? *

- Evet
- Hayır
- Uygulaması Yok

4. Tutarlılık ve Standartlar

Kullanıcılar, farklı kelime, durum ve hareketlerin aynı şeyi ifade edip etmediğini bilmek zorunda değildirler. Ortak bir tutarlılık tüm sistemde geçerli olmalıdır.

1-Bir ekranda büyük harflerin sıklıkla kullanımından kaçınılıyor mu? *

- Evet
- Hayır
- Uygulaması Yok

2-Bütün ikonlar etiketli (yani yazıları var mı) mi? *

- Evet
- Hayır
- Uygulaması Yok

3-Her pencerede yatay ve dikey kaydırma (scrollling) mümkün mü? *

- Evet
- Hayır
- Uygulaması Yok

4-Menü yapısı, görev yapısı ile uyuşuyor mu? *

yani dosya menüsü dosya ile ilgili işlemleri mi yapıyor?

- Evet
- Hayır
- Uygulaması Yok

5-Menüler dikey olarak sunulmuş mudur? *

- Evet
- Hayır
- Uygulaması Yok

6-Eğer "çıkış" bir menü seçeneği ise listenin her zaman en altında yer almakta mıdır? *

- Evet
- Hayır
- Uygulaması Yok

7-Menü başlıkları ortaya veya sola hizalı mıdır? *

- Evet
- Hayır
- Uygulaması Yok

8-Alan etiketleri ve alanlar görsel olarak ayırt edilebiliyor mu? *

- Evet
- Hayır
- Uygulaması Yok

9-Aşağıdaki dikkat çekme teknikleri itina ile kullanılmış mı? *

- ☐ Yazı büyüklüğü: en fazla 10 punto
- ☐ Yazı Tipi: en fazla 3 çeşit
- ☐ Renk: en fazla 4
- ☐ Ses: Düzenli olumlu geribildirim için yumuşak tonlar, nadir kritik durumlar için sert tonlar

10-Önemli bilgiler uyarı mesajlarının başında olacak şekilde yerleştirilmiş midir? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

11-Sistem nesneleri (düğmeler, bağlantılar, sekmeler, alanlar) tutarlı bir şekilde isimlendirilmiş midir? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma

Hata geri dönütleri, sade bir dilde (kodsuz) olmalı, sorunu açıklamalı ve yapıcı çözüm önerisi sunmalıdır.

1-Bir hatayı bildirmek için ses kullanılıyor mu? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

2-Hatırlatıcılar, işlemin kullanıcının kontrolünde olduğunu ima etmekte midir? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

3-Hatırlatıcılar, kısa, net ve anlaşılır mı? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

4-Hata mesajları kullanıcının değil, sistemin sorumlu olduğunu belirtecek şekilde ifade edilmekte midir? *

- Evet
- Hayır
- Uygulaması Yok

5-Hata mesajları gramer açısından doğru mudur? *

- Evet
- Hayır
- Uygulaması Yok

6-Hata mesajları sorunun nedeni hakkında bilgiler sunuyor mu? *

- Evet
- Hayır
- Uygulaması Yok

7-Hata mesajları, kullanıcıya hatayı düzeltmek için hangi hareketi yapması gerektiğini belirtiyor mu? *

- Evet
- Hayır
- Uygulaması Yok

6. Hataları Önleme

Kullanıcıların bir hata mesajı ile karşılaşması yerine, dikkatli bir tasarım ile hatanın oluşması önlenmelidir.

1-Menü seçenekleri mantıksal, birbirinden farklı ve birbirini kapsamayan şekilde midir? *

- Evet
- Hayır
- Uygulaması Yok

2-Veri girişleri mümkün olduğu kadarıyla büyük ve küçük harfe duyarlı olarak girilebiliyor mu? *

- Evet
- Hayır
- Uygulaması Yok

4-Kullanıcıların dikkatini çekmek için birbirine zıt renkler kullanılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

5-Herhangi bir maddenin seçilmiş olduğunu gösteren birbirine zıt görsel ipuçlarından yararlanılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

6-Anlamlı grupları belirlemek için çerçeveler kullanılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

7-İlişkili elemanları gruplamak için aynı renkler kullanılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

8-Renk tonları sistem boyunca tutarlı bir şekilde kullanılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

8. Esneklik ve Verimlilik

Acemi kullanıcılar tarafından görülemeyen hızlandırıcılar kullanılmalıdır. Genellikle sistemin deneyimli ve deneyimsiz kullanıcıları farklı kullanım davranışı gösterirler. Her iki gruba da hitap etmek için, uzman kullanıcılar için etkileşimi hızlandırıcı yöntemler kullanılmalıdır. Kullanıcılara sık kullandıkları fonksiyonları isteklerine göre ayarlayabilmeleri için imkânlar sunulmalıdır.

1-Eğer sistem acemi ve profesyonel kullanıcıların ikisini de destekliyor ise detaylı hata mesajları farklı düzeylerde sağlanmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

2-Sistem acemi kullanıcılara, her komutun en basit, en yaygın biçimini girmeyi ve profesyonel kullanıcılara parametreleri eklemeleri için izin veriyor mu? *

- Evet
- Hayır
- Uygulaması Yok

3-Sistem sıkça kullanılan komutlar için fonksiyon tuşları/komutları sağlıyor mu? *

- Evet
- Hayır
- Uygulaması Yok

4-Kullanıcılar, diyalog kutularında, hem doğrudan diyalog kutusunu tıklama hem de klavyedeki kısa yol tuşlarını kullanma seçeneklerine sahip midir? *

- Evet
- Hayır
- Uygulaması Yok

9. Estetik ve Sade Tasarım

Sistem alakasız ve az ihtiyaç duyulan bilgiyi içermemelidir.

1-Her bir ikon arkaplandan ayırt edilebiliyor mu? *

- Evet
- Hayır
- Uygulaması Yok

2-Anlamlı madde grupları birbirlerinden ayrılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

3-Her bir veri giriş ekranı kısa, basit, net ve ayırt edilebilir bir başlığa sahip midir? *

- Evet
- Hayır
- Uygulaması Yok

4-Alan başlıkları kısa, tanıdık ve açıklayıcı mıdır? *

- Evet
- Hayır
- Uygulaması Yok

5-Menü başlıkları kısa fakat iletişimi sağlamak için yeterli uzunlukta mıdır? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

6-Sistem, veri girmek için kutu dışında açılır kutu, radyo butonu gibi başka seçenekler sunuyor mu? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

10. Yardım ve Dokümantasyon

Dokümantasyon olmadan sistemi kullanabilmek daha tercih edilir olmasına rağmen, kullanıcıya dokümantasyon ve yardım servisi sunmak gerekli olabilir. Yardım sisteminde gereken bilgiyi aramak kolay olmalı, yardım dokümanı kullanıcının görevine odaklı olmalı, çözümler listelerken somut adımları göstermeli ve çok büyük olmamalıdır.

1-Çevrim içi bilgilendirmeler (online yardım gibi) görsel olarak ayırt edilebilir durumda mıdır? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

2-Bilgilendirmeler kullanıcının yapacağı işlemlere uygun olarak sıralanmış mıdır? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

3-Yardım işlevi görünür mü? Örneğin YARDIM adında bir düğme veya özel bir menü mevcut mudur? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

4-Sistemin sunduğu yardım için aşağıdakilerden hangisi geçerlidir? *

- ☐ Bilgiler anlamlı mı?
- ☐ Amaç odaklı mı? (Bu programla ne yapabilirim?)
- ☐ İşlevsel mi? (Bu görevi nasıl yapacağım?)
- ☐ Yorumlayıcı mı? (Bu neden oldu?)
- ☐ Gezinilebilir mi? (Ben neredeyim?)
- ☐ Olaya/nedene duyarlı yardım mevcut mu?

5-Kullanıcı, var olan detay seviyesini değiştirebiliyor mu? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

6-Yardım sistemine kolayca ulaşılabiliyor mu veya bu bölümden kolayca çıkılabiliyor mu? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

11.Yetenekler

Sistem kullanıcının yeteneklerinin ve uzmanlığının gelişmesine ve fazlalaşmasına olanak sağlamalıdır.

1-Pencere işlemlerinin yapılması ve öğrenilmesi kolay mıdır? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

2-Kullanıcılar hareketlerin cevaplayıcısından çok başlatıcısı mı? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

3-Sistem kullanıcılar için veri transferi yapıyor mu? *

Açma ya da saklama yapabiliyor mu?

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

4-Kullanıcılar, bir alan içerisinde ileri ve geri hareket edebiliyorlar mı? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

5-Fonksiyonelliği desteklemeye yetecek kadar sayıda, ancak aramayı ve bulmayı zorlaştırmayacak kadar yeterli işlev komutu (ekranda) var mı? *

- Evet
- Hayır
- Uygulaması Yok

6-İşlev komutları (aç, sakla, yazdır gibi) sık yapılan önemli işlemler için kullanılmakta mıdır? *

- Evet
- Hayır
- Uygulaması Yok

12. Kullanıcı ile Seviyeli bir İletişim

Kullanıcının sistemle etkileşimi onun iş yaşamındaki kaliteyi arttırmalıdır. Kullanıcıya saygılı davranmak, tasarım görünüş ve işlemesi bakımından tatmin edici olmalıdır.

1-Her bir ikon (simge), bir ikon grubunun uyumlu bir öğesi midir? *

- Evet
- Hayır
- Uygulaması Yok

2-İkonların tasarımında aşırı detaydan sakınılmış mıdır? *

- Evet
- Hayır
- Uygulaması Yok

3-Renk kullanımı dikkatle kullanılmış mı? *

- Evet
- Hayır
- Uygulaması Yok

4-Renkler özellikle dikkat çekme, iletişim, organizasyon, değişen durumları gösterme ve ilişkileri açıklama amaçlı kullanılmış mı? *

- Evet
- Hayır
- Uygulaması Yok

5-En sık kullanılan işlev komutları/yönergeleri, en ulaşılabilir pozisyonda mı? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

6-Sistem, veri giriş alanlarına bir bölümü girilmiş verileri tamamlıyor mu? *

- ○ Evet
- ○ Hayır
- ○ Uygulaması Yok

Ek 2- BT Öğretmenlerine Gönderilen Örnek E-Postalar

Microsoft Kodu Game Lab yazılımını inceleyecek kişiler için gönderilen örnek bir e-postadır.

Değerli **FATMA AKYOL** hocam,

Telefonda sizlere kendimi tanıtıp programlama öğretimi alanında yaptığım tez için destek istemiştim. Öncelikle kabul ettiğinizden ötürü teşekkür ederim. Programlama öğretiminde kullanılan popüler 4 programı (aracı) belirledim. Bu programların Kullanılabilirlik ve Tasarımlarını araştırmaktayım. Sizden bu programlardan sadece birini (1) bilgisayarınıza kurup incelemeniz istenmektedir. İnceleyeceğiniz program aşağıda verilmiştir.

KODU GAME LAB

Çalışmamızda öğrenciler kullanılmayacaktır. Programlama öğretimi yapan sizlerin uzman görüşlerini alıyorum. Yukarıdaki programı ister hazırladığım Google Drive'dan isterseniz direkt web sitelerinden çekebilirsiniz.

G.Drive:
https://drive.google.com/file/d/0B1Kdc9sflBb2MHNrNjE3bUhLWk0/view?usp=sharing
Direkt İndirme: http://www.microsoft.com/en-us/download/details.aspx?id=10056
Sizlere yardımcı ve bilgilendirici olması amacıyla bu programlama öğretim aracı ile ilgili bir video (Youtube) hazırladım. Arzu ederseniz izleyebilirsiniz.

https://www.youtube.com/watch?v=tnMlUQKVS7U
Programı bir müddet kullandıktan sonra aşağıdaki anketi doldurmanız beklenmektedir. Anket, Google Documents'da hazırlanmıştır. Mayıs sonuna kadar bu aracı değerlendirip tarafıma gönderirseniz müteşekkir olurum.
Anket
https://docs.google.com/forms/d/11jADlUfPqva_UPrDj53qWYSifSxr3XWVbCK_napBgcQ/viewform?usp=send_form

Verdiğiniz desteğin tezime yardımcı olması yanı sıra hem BTY müfredatına hem de sizin programlama öğretimi araçları bilginize katkısının olacağını düşünüyorum.

Çalışma ile ilgili sormak istediğiniz herhangi bir soru olursa e-posta gönderebilirsiniz
Çalışmalarınızda başarılar dilerim.

Microsoft Small Basic yazılımını inceleyecek kişiler için gönderilen örnek bir e-postadır.

Değerli **FATİH SOYSAL** hocam,

Telefonda sizlere kendimi tanıtıp programlama öğretimi alanında yaptığım tez için destek istemiştim. Öncelikle kabul ettiğinizden ötürü teşekkür ederim. Programlama öğretiminde kullanılan popüler 4 programı (aracı) belirledim. Bu programların Kullanılabilirlik ve Tasarımlarını araştırmaktayım. Sizden bu programlardan sadece birini (1) bilgisayarınıza kurup incelemeniz istenmektedir. İnceleyeceğiniz program aşağıda verilmiştir.

SMALL BASIC

Çalışmamızda öğrenciler kullanılmayacaktır. Programlama öğretimi yapan sizlerin uzman görüşlerini alıyorum. Yukarıdaki programı ister hazırladığım Google Drive'dan isterseniz direkt web sitelerinden çekebilirsiniz.

G.Drive:
https://drive.google.com/file/d/0B1Kdc9sflBb2cFdQUXRBZTRleXc/view?usp=sharing
Direkt İndirme: https://www.microsoft.com/tr-TR/download/details.aspx?id=46392

Sizlere yardımcı ve bilgilendirici olması amacıyla bu programlama öğretim aracı ile ilgili bir video (Youtube) hazırladım. Arzu ederseniz izleyebilirsiniz.

https://www.youtube.com/watch?v=iwSN6yEdxQY
Programı bir müddet kullandıktan sonra aşağıdaki anketi doldurmanız beklenmektedir. Anket, Google Documents'da hazırlanmıştır. Mayıs sonuna kadar bu aracı değerlendirip tarafıma gönderirseniz müteşekkir olurum.
Anket
https://docs.google.com/forms/d/11jADlUfPqva_UPrDj53qWYSifSxr3XWVbCK_napBgcQ/viewform?usp=send_form

Verdiğiniz desteğin tezime yardımcı olması yanı sıra hem BTY müfredatına hem de sizin programlama öğretimi araçları bilginize katkısının olacağını düşünüyorum.

Çalışma ile ilgili sormak istediğiniz herhangi bir soru olursa e-posta gönderebilirsiniz
Çalışmalarınızda başarılar dilerim.

Scratch yazılımını inceleyecek kişiler için gönderilen örnek bir e-postadır.

Değerli **RIZA BOZLU** hocam,

Telefonda sizlere kendimi tanıtıp programlama öğretimi alanında yaptığım tez için destek istemiştim. Öncelikle kabul ettiğinizden ötürü teşekkür ederim. Programlama öğretiminde kullanılan popüler 4 programı (aracı) belirledim. Bu programların Kullanılabilirlik ve Tasarımlarını araştırmaktayım. Sizden bu programlardan sadece birini (1) bilgisayarınıza kurup incelemeniz istenmektedir. İnceleyeceğiniz program aşağıda verilmiştir.

SCRATCH

Çalışmamızda öğrenciler kullanılmayacaktır. Programlama öğretimi yapan sizlerin uzman görüşlerini alıyorum. Yukarıdaki programı ister hazırladığım Google Drive'dan isterseniz direkt web sitelerinden çekebilirsiniz.

G.Drive:https://drive.google.com/file/d/0B1Kdc9sflBb2LWhGS1VEU0daUUU/view?usp=sharing

Direkt İndirme:https://scratch.mit.edu/scratch_1.4/

Sizlere yardımcı ve bilgilendirici olması amacıyla bu programlama öğretim aracı ile ilgili bir video (Youtube) hazırladım. Arzu ederseniz izleyebilirsiniz.

https://www.youtube.com/watch?v=z7dRQgl2WII

Programı bir müddet kullandıktan sonra aşağıdaki anketi doldurmanız beklenmektedir. Anket, Google Documents'da hazırlanmıştır. Mayıs sonuna kadar bu aracı değerlendirip tarafıma gönderirseniz müteşekkir olurum.

Anket
https://docs.google.com/forms/d/11jADlUfPqva_UPrDj53qWYSifSxr3XWVbCK_napBgcQ/viewform?usp=send_form

Verdiğiniz desteğin tezime yardımcı olması yanı sıra hem BTY müfredatına hem de sizin programlama öğretimi araçları bilginize katkısının olacağını düşünüyorum.

Çalışma ile ilgili sormak istediğiniz herhangi bir soru olursa e-posta gönderebilirsiniz Çalışmalarınızda başarılar dilerim.

Robomind yazılımını inceleyecek kişiler için gönderilen örnek bir e-postadır.

Değerli **AHMET BEŞİKTEPE** hocam,

Telefonda sizlere kendimi tanıtıp programlama öğretimi alanında yaptığım tez için destek istemiştim. Öncelikle kabul ettiğinizden ötürü teşekkür ederim. Programlama öğretiminde kullanılan popüler 4 programı (aracı) belirledim. Bu programların Kullanılabilirlik ve Tasarımlarını araştırmaktayım. Sizden bu programlardan sadece birini (1) bilgisayarınıza kurup incelemeniz istenmektedir. İnceleyeceğiniz program aşağıda verilmiştir.

ROBOMIND

Çalışmamızda öğrenciler kullanılmayacaktır. Programlama öğretimi yapan sizlerin uzman görüşlerini alıyorum. Yukarıdaki programı ister hazırladığım Google Drive'dan isterseniz direkt web sitelerinden çekebilirsiniz.

G.Drive:https://drive.google.com/file/d/0B1Kdc9sflBb2a2l4RkJLaVlEUkE/view?usp=sharing

Direkt İndirme: http://www.robomind.net/rmnet/download/windows

Sizlere yardımcı ve bilgilendirici olması amacıyla bu programlama öğretim aracı ile ilgili bir video (Youtube) hazırladım. Arzu ederseniz izleyebilirsiniz.

https://www.youtube.com/watch?v=zwbBNFcINUo

Programı bir müddet kullandıktan sonra aşağıdaki anketi doldurmanız beklenmektedir. Anket, Google Documents'da hazırlanmıştır. Mayıs sonuna kadar bu aracı değerlendirip tarafıma gönderirseniz müteşekkir olurum.

Anket

https://docs.google.com/forms/d/11jADlUfPqva_UPrDj53qWYSifSxr3XWVbCK_napBgcQ/viewform?usp=send_form

Verdiğiniz desteğin tezime yardımcı olması yanı sıra hem BTY müfredatına hem de sizin programlama öğretimi araçları bilginize katkısının olacağını düşünüyorum.

Çalışma ile ilgili sormak istediğiniz herhangi bir soru olursa e-posta gönderebilirsiniz Çalışmalarınızda başarılar dilerim.

DEMOGRAFİK BİLGİLER

YANITLAR

1-Cinsiyetiniz

Kadın	Erkek	Toplam
35	57	92

3-Hizmet yılınız nedir?

1--3 yıl	4--6 yıl	6--9 yıl	10--12 yıl	13--15 yıl	16--18 yıl
11	26	34	19	1	1

4-Hangi programlama öğretimi aracını kullandınız?

Scratch	Small Basic	Kodu Game	Robomind	Toplam
21	24	24	23	92

5-Programlama bilginizi derecelendirseydiniz kendinize kaç verirdiniz?

1 2 3 4 5 6 7 8 9 10

○ ○ ○ ○ ○ ○ ○ ○ ○ ○

2	3	4	5	6	7	8	9
1	4	6	18	10	28	18	7

6-Programlama öğretiminde görsel araçların kullanılmasını nasıl değerlendiriyorsunuz?

	Scratch	Small Basic	Kodu Game	Robomind	Toplam
Çok faydasının dokunacağı kanaatindeyim.	20	19	19	20	**78**
Kararsızım.	0	1	3	1	**5**
Yararlı olacağını düşünmüyorum	1	4	2	2	**9**

7-Kullandığınız programlama öğretimi aracının hangi yaş grubuna hitap ettiğini düşünüyorsunuz?

	Scratch	Small Basic	Kodu Game	Robomind	Toplam
İlkokul 3-4	0	0	1	0	**1**
İlkokul tüm sınıflar	0	1	1	1	**3**
Ortaokul 5-6	9	1	4	9	**23**
Ortaokul 7-8	1	8	4	2	**15**
İlkokul ve Ortaokul	2	0	3	1	**6**
Ortaokul tüm sınıflar	7	7	8	7	**29**
Lise 9-10	0	1	1	2	**4**
Lise 11-12	0	1	0	0	**1**
Ortaokul ve Lise	1	2	2	1	**6**
Lise tüm sınıflar	1	3	0	0	**4**

8-Sizce bu tür araçlarda Kullanılabilirlik veya Tasarımdan hangisine daha çok dikkat edilmelidir?

	Scratch	Small Basic	Kodu Game	Robomind	Toplam
Kullanılabilirlik	1	4	1	2	**8**
Tasarım	1	1	2	0	**4**
Her ikiside	19	19	21	21	**80**

9-Kullandığınız aracın Kullanılabilirliğini nasıl derecelendirirsiniz?

1 2 3 4 5

Çok kötü ○ ○ ○ ○ ○ Çok iyi

	Scratch	Small Basic	Kodu Game	Robomind	Toplam
2	0	0	1	0	**1**
3	3	10	4	2	**19**
4	16	12	8	20	**56**
5	2	2	11	1	**16**

10-Kullandığınız aracın Tasarımını nasıl derecelendirirsiniz?

1 2 3 4 5

Çok kötü ○ ○ ○ ○ ○ Çok iyi

	Scratch	Small Basic	Kodu Game	Robomind	Toplam
2	0	0	2	1	**3**
3	3	10	5	7	**25**
4	16	11	8	12	**47**
5	2	3	9	3	**17**

SORULAR

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
1. Sistem Durumunun Görünürlüğü		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Tüm sistemde gösterge (ikon) tasarım ve stil şekilleri tutarlı mı?	17	2	2	22	1	1	23	1	0	20	2	1
2	Her menüdeki bilgilendirmeler, hatırlatmalar ve hata mesajları sistem içerisinde aynı yerde görünmekte midir?	7	4	10	19	3	2	18	4	2	22	0	1
3	Eğer hata mesajları açılan bir alan içerisinde gösteriliyor ise, kullanıcı, hatalı olan alanı görebilmekte midir?	5	7	9	19	2	3	13	2	9	19	2	2
4	Nesnelerin seçildiğini veya hareket ettirildiğini gösteren görsel geribildirim mevcut mudur?	18	2	1	10	6	8	22	2	0	19	4	0
5	Grafik arayüzlü menülerde, hangi seçeneğin seçildiği açık şekilde belli oluyor mu?	19	2	0	14	3	7	21	3	0	18	1	4
	Yanıtların Toplamı	**66**	**17**	**22**	**84**	**15**	**21**	**97**	**12**	**11**	**98**	**9**	**8**

2. Sistem ve Gerçek Dünyanın Eşleşmesi		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Sistem içerisinde yer alan göstergeler (ikon), kullanıcı için somut ve tanıdık mıdır?	20	1	0	21	3	0	19	4	1	22	1	0
2	Eğer görsel bir ipucu olarak kullanılan şekil varsa, bu kültürel geleneklerle eşleşiyor mu?	10	6	5	12	8	4	14	9	1	13	6	4
3	Seçilen renkler, renk kodları ile ilgili genel beklentileri karşılıyor mu?	16	5	0	21	1	1	17	5	2	18	2	3
4	Hatırlatmalar, sistem içerisinde gerekli hareketleri açıklarken, kullanılan mesaj açıklanan harekete uygun mudur?	13	1	7	20	0	4	22	0	2	23	0	0
5	Menü başlıkları dilbilgisi yönünden birbirleri ile tutarlı mıdır?	18	3	0	22	0	2	20	4	0	21	2	0
6	Sistem kullanıcı jargonunu kullanıp ve bilgisayar jargonundan sakınmakta mıdır?	13	5	3	9	11	4	8	12	4	12	10	1
7	Komut isimleri anlamlı mıdır?	18	3	0	24	0	0	20	4	0	23	0	0
	Yanıtların Toplamı	**108**	**24**	**15**	**129**	**23**	**15**	**120**	**38**	**10**	**132**	**21**	**8**

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
3. Kullanıcı Kontrol ve Özgürlüğü		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Seyrek (az) olarak kullanılan görevlerin hatırlanması kolay mıdır?	14	7	0	14	7	3	17	6	1	16	7	0
2	Açılır pencere olan sistemlerde, kullanıcılar için pencereler arası geçiş kolay mıdır?	9	2	10	18	3	3	19	3	2	20	1	2
3	Herhangi bir işlem hareketi, veri girişi veya birçok diğer işlem hareketleri için "geri al" fonksiyonu bulunmakta mıdır?	9	11	1	22	2	0	20	3	1	22	1	0
4	Kullanıcılar, devam eden işlemleri iptal edebiliyorlar mı?	19	1	1	19	3	2	21	2	1	22	0	1
5	Sistem, kopyalayarak veya varolan veriyi değiştirerek kullanıcının veri giriş süresini azaltıyor mu?	16	3	2	18	4	2	16	4	4	18	4	1
6	Kullanıcıların menü öğelerini tıklayarak veya klavye kısayolunu kullanarak seçme olanağı var mı?	15	4	2	22	2	0	20	3	1	21	1	1
7	Menüler iç içe olmaktan ziyade daha geniş (bir menüde birden fazla öğe,ribon tarzı) mi?	19	1	1	18	3	3	16	7	1	19	4	0
	Yanıtların Toplamı	**101**	**29**	**17**	**131**	**24**	**13**	**129**	**28**	**11**	**138**	**18**	**5**

4. Tutarlılık ve Standartlar		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Bir ekranda büyük harflerin sıklıkla kullanımından kaçınılıyor mu?	16	0	5	18	4	2	17	3	4	14	7	2
2	Bütün ikonlar etiketli (yani yazıları var mı) mi?	14	6	1	23	0	1	18	6	0	19	4	0
3	Her pencerede yatay ve dikey kaydırma (scrollling) mümkün mü?	10	7	4	20	3	1	6	9	9	14	8	1
4	Menü yapısı, görev yapısı ile uyuşuyor mu?	20	1	0	22	0	4	21	2	1	19	3	1
5	Menüler dikey olarak sunulmuş mudur?	17	4	0	6	14	4	10	11	3	11	12	0
6	Eğer "çıkış" bir menü seçeneği ise listenin her zaman en altında yer almakta mıdır?	11	5	5	3	11	10	9	10	5	12	8	3
7	Menü başlıkları ortaya veya sola hizalı mıdır?	19	2	0	19	3	2	18	6	0	22	1	0
8	Alan etiketleri ve alanlar görsel olarak ayırt edilebiliyor mu?	18	2	1	22	1	1	22	2	0	18	4	1
9	Aşağıdaki dikkat çekme teknikleri itina ile kullanılmış mı?	11	0	0	16	0	0	16	0	0	14	0	0
10	Önemli bilgiler uyarı mesajlarının başında olacak şekilde yerleştirilmiş midir?	8	5	8	17	6	1	19	3	2	17	2	4
11	Sistem nesneleri (düğmeler, bağlantılar, sekmeler, alanlar) tutarlı bir şekilde isimlendirilmiş midir?	17	4	0	21	1	2	24	0	0	23	0	0
	Yanıtların Toplamı	**161**	**36**	**24**	**187**	**43**	**28**	**180**	**52**	**24**	**183**	**49**	**12**

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Bir hatayı bildirmek için ses kullanılıyor mu?	4	9	8	6	18	0	17	2	5	7	14	2
2	Hatırlatıcılar, işlemin kullanıcının kontrolünde olduğunu ima etmekte midir?	6	6	9	16	3	5	17	1	6	17	4	2
3	Hatırlatıcılar, kısa, net ve anlaşılır mı?	5	5	11	18	2	4	18	2	4	20	1	2
4	Hata mesajları kullanıcının değil, sistemin sorumlu olduğunu belirtecek şekilde ifade edilmekte midir?	2	6	13	6	12	6	8	4	12	16	5	2
5	Hata mesajları gramer açısından doğru mudur?	3	4	14	21	3	0	12	2	10	20	3	0
6	Hata mesajları sorunun nedeni hakkında bilgiler sunuyor mu?	2	6	13	19	5	0	8	5	11	16	7	0
7	Hata mesajları, kullanıcıya hatayı düzeltmek için hangi hareketi yapması gerektiğini belirtiyor mu?	1	8	12	9	13	2	8	4	12	12	9	2
	Yanıtların Toplamı	**23**	**44**	**80**	**95**	**56**	**17**	**88**	**20**	**60**	**108**	**43**	**10**

6. Hataları Önleme		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Menü seçenekleri mantıksal, birbirinden farklı ve birbirini kapsamayan şekilde midir?	14	5	4	17	4	3	17	6	1	19	4	0
2	Veri girişleri mümkün olduğu kadarıyla büyük ve küçük harfe duyarlı olarak girilebiliyor mu?	14	7	0	15	6	3	11	3	10	13	10	0
3	Ciddi sonuçları olabilecek gösterge/düğmeler ulaşılması zor bir yerde midir?	5	13	3	2	19	3	7	13	4	3	18	2
4	Sistem, kullanıcıların mümkün olabildiğince hata yapmasını önleyebiliyor mu?	10	7	4	17	7	0	20	3	1	16	7	0
5	Sistem, kullanıcıları sonuçları ciddi ve yıkıcı olabilecek hatalar yapmak üzere iken onları uyarıyor mu?	3	7	11	8	6	10	7	7	10	8	11	4
	Yanıtların Toplamı	**46**	**39**	**22**	**59**	**42**	**19**	**62**	**32**	**26**	**59**	**50**	**6**

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
7. Hatırlama Yerine Tanıma		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Veri gösterimi ekranın sol üst köşesinden başlıyor mu?	19	1	1	18	4	2	21	1	2	19	4	0
2	Hatırlatıcılar, ipuçları ve mesajlar ekranda gözlerin genellikle baktığı yerlere yerleştirilmiş mi?	13	3	5	21	2	1	21	3	0	16	6	1
3	Metin alanlarının çevrelerinde boşluk var mı?	15	5	1	21	2	1	20	3	1	22	0	1
4	Kullanıcıların dikkatini çekmek için birbirine zıt renkler kullanılmış mıdır?	18	2	1	18	6	0	19	5	0	20	3	0
5	Herhangi bir maddenin seçilmiş olduğunu gösteren birbirine zıt görsel ipuçlarından yararlanılmış mıdır?	13	7	1	17	6	1	23	1	0	18	4	1
6	Anlamlı grupları belirlemek için çerçeveler kullanılmış mıdır?	18	1	2	15	4	4	19	4	1	16	5	2
7	İlişkili elemanları gruplamak için aynı renkler kullanılmış mıdır?	20	1	0	18	2	4	21	3	0	17	5	1
8	Renk tonları sistem boyunca tutarlı bir şekilde kullanılmış mıdır?	18	3	0	23	1	1	20	3	1	21	1	1
	Yanıtların Toplamı	**134**	**23**	**11**	**151**	**27**	**14**	**164**	**23**	**5**	**149**	**28**	**7**
8. Esneklik ve Verimlilik		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Eğer sistem acemi ve profesyonel kullanıcıların ikisini de destekliyor ise detaylı hata mesajları farklı düzeylerde sağlanmış mıdır?	5	6	10	4	17	3	6	10	8	11	12	0
2	Sistem acemi kullanıcılara, her komutun en basit, en yaygın biçimini girmeyi ve profesyonel kullanıcılara parametreleri eklemeleri için izin veriyor mu?	11	4	8	17	6	1	15	8	1	15	7	1
3	Sistem sıkça kullanılan komutlar için fonksiyon tuşları/komutları sağlıyor mu?	7	8	8	13	7	4	14	8	2	14	9	0
4	Kullanıcılar, diyalog kutularında, hem doğrudan diyalog kutusunu tıklama hem de klavyedeki kısa yol tuşlarını kullanma seçeneklerine sahip midir?	6	10	5	12	4	8	16	5	3	15	6	2
	Yanıtların Toplamı	**29**	**28**	**31**	**46**	**34**	**16**	**51**	**31**	**14**	**55**	**34**	**3**

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
9. Estetik ve Sade Tasarım		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Her bir ikon arkaplandan ayırt edilebiliyor mu?	18	3	0	22	1	1	22	2	0	20	3	0
2	Anlamlı madde grupları birbirlerinden ayrılmış mıdır?	19	2	0	21	0	3	22	2	0	21	2	0
3	Her bir veri giriş ekranı kısa, basit, net ve ayırt edilebilir bir başlığa sahip midir?	18	2	1	22	1	1	21	2	1	20	1	2
4	Alan başlıkları kısa, tanıdık ve açıklayıcı mıdır?	19	2	0	22	1	1	20	3	1	22	1	0
5	Menü başlıkları kısa fakat iletişimi sağlamak için yeterli uzunlukta mıdır?	18	3	0	20	2	2	20	3	1	22	0	1
6	Sistem, veri girmek için kutu dışında açılır kutu, radyo butonu gibi başka seçenekler sunuyor mu?	8	10	3	9	11	4	14	6	4	11	10	2
	Yanıtların Toplamı	**100**	**22**	**4**	**116**	**16**	**12**	**119**	**18**	**7**	**116**	**17**	**5**

10. Yardım ve Dokümantasyon		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Çevrim içi bilgilendirmeler (online yardım gibi) görsel ola-rak ayırt edilebilir durumda mıdır?	16	4	1	7	8	9	14	4	6	19	3	1
2	Bilgilendirmeler kullanıcının yapacağı işlemlere uygun olarak sıralanmış mıdır?	13	6	2	15	4	5	19	3	2	21	2	0
3	Yardım işlevi görünür mü? Örneğin YARDIM adında bir düğme veya özel bir menü mevcut mudur?	19	2	0	9	10	5	20	2	2	23	0	0
4	Sistemin sunduğu yardım için aşağıdakilerden hangisi geçerlidir?	9	0	0	11	0	0	11	0	0	12	0	0
5	Kullanıcı, var olan detay seviyesini değiştirebiliyor mu?	6	11	4	4	17	3	13	7	4	10	12	1
6	Yardım sistemine kolayca ulaşılabiliyor mu veya bu bölümden kolayca çıkılabiliyor mu?	18	3	0	9	3	12	20	4	0	22	1	0
	Yanıtların Toplamı	**81**	**26**	**7**	**55**	**42**	**34**	**97**	**20**	**14**	**107**	**18**	**2**

KONTROL ARACI SORULARINA YAZILIMLAR İÇİN VERİLEN YANITLAR		Scratch			Small Basic			Kodu Game			Robomind		
11.Yetenekler		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Pencere işlemlerinin yapılması ve öğrenilmesi kolay mıdır?	19	1	1	23	1	0	20	3	1	23	0	0
2	Kullanıcılar hareketlerin cevaplayıcısından çok başlatıcısı mı?	20	1	0	18	4	2	21	3	0	23	0	0
3	Sistem kullanıcılar için veri transferi yapıyor mu?	18	1	2	18	4	2	20	4	0	16	4	3
4	Kullanıcılar, bir alan içerisinde ileri ve geri hareket edebiliyorlar mı?	15	4	2	14	6	4	19	1	4	22	0	1
5	Fonksiyonelliği desteklemeye yetecek kadar sayıda, ancak aramayı ve bulmayı zorlaştırmayacak kadar yeterli işlev komutu (ekranda) var mı?	18	3	0	15	8	1	18	6	0	22	1	0
6	İşlev komutları (aç, sakla, yazdır gibi) sık yapılan önemli işlemler için kullanılmakta mıdır?	17	4	0	21	2	1	16	4	4	19	4	0
	Yanıtların Toplamı	**107**	**14**	**5**	**109**	**25**	**10**	**114**	**21**	**9**	**125**	**9**	**4**

12. Kullanıcı ile Seviyeli Bir İletişim		Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok	Evet	Hayır	U.Yok
1	Her bir ikon (simge), bir ikon grubunun uyumlu bir öğesi midir?	19	2	0	22	2	0	19	5	0	22	1	0
2	İkonların tasarımında aşırı detaydan sakınılmış mıdır?	19	2	0	22	2	0	20	3	2	22	1	0
3	Renk kullanımı dikkatle kullanılmış mı?	19	2	0	23	1	0	20	4	0	22	1	0
4	Renkler özellikle dikkat çekme, ileti-şim, organizasyon, değişen durumları gösterme ve ilişkileri açıklama amaçlı kullanılmış mı?	18	3	0	21	2	1	20	4	0	20	3	0
5	En sık kullanılan işlev komutları/yönergeleri, en ulaşılabilir pozisyonda mı?	18	2	1	21	2	1	21	3	0	23	0	0
6	Sistem, veri giriş alanlarına bir bölü-mü girilmiş verileri tamamlıyor mu?	12	6	3	19	4	1	21	3	0	12	10	1
	Yanıtların Toplamı	**105**	**17**	**4**	**128**	**13**	**3**	**121**	**22**	**2**	**121**	**16**	**1**

Ek 4- Problemlerin Yoğunlaştığı Sorular ve Yazılımlara Göre Dağılımı

SORUNLARIN YOĞUNLAŞTIĞI SORULAR ve SAYILARI		Scratch		Small Basic		Kodu Game		Robomind	
1. Sistem Durumunun Görünürlüğü		Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok
2	Her menüdeki bilgilendirmeler, hatırlatmalar ve hata mesajları sistem içerisinde aynı yerde görünmekte midir?		10						
3	Eğer hata mesajları açılan bir alan içerisinde gösteriliyor ise, kullanıcı, hatalı olan alanı görebilmekte midir?		9						
2. Sistem ve Gerçek Dünyanın Eşleşmesi									
2	Eğer görsel bir ipucu olarak kullanılan şekil varsa, bu kültürel geleneklerle eşleşiyor mu?					9			
6	Sistem kullanıcı jargonunu kullanıp ve bilgisayar jargonundan sakınmakta mıdır?					12			
3. Kullanıcı Kontrol ve Özgürlüğü									
2	Açılır pencere olan sistemlerde, kullanıcılar için pencereler arası geçiş kolay mıdır?		10						
3	Herhangi bir işlem hareketi, veri girişi veya birçok diğer işlem hareketleri için "geri al" fonksiyonu bulunmakta mıdır?	11							
4. Tutarlılık ve Standartlar									
5	Menüler dikey olarak sunulmuş mudur?					11			
6	Eğer "çıkış" bir menü seçeneği ise listenin her zaman en altında yer almakta mıdır?				10	10			
5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma									
1	Bir hatayı bildirmek için ses kullanılıyor mu?			18					
2	Hatırlatıcılar, işlemin kullanıcının kontrolünde olduğunu ima etmekte midir?		9						
3	Hatırlatıcılar, kısa, net ve anlaşılır mı?		11						
4	Hata mesajları kullanıcının değil, sistemin sorumlu olduğunu belirtecek şekilde ifade edilmekte midir?		13	12			12		
5	Hata mesajları gramer açısından doğru mudur?		14						
6	Hata mesajları sorunun nedeni hakkında bilgiler sunuyor mu?		13				11		
7	Hata mesajları, kullanıcıya hatayı düzeltmek için hangi hareketi yapması gerektiğini belirtiyor mu?		12	13			12		

SORUNLARIN YOĞUNLAŞTIĞI SORULAR ve SAYILARI		Scratch		Small Basic		Kodu Game		Robomind	
6. Hataları Önleme		Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok
2	Veri girişleri mümkün olduğu kadarıyla büyük ve küçük harfe duyarlı olarak girilebiliyor mu?						10	10	
3	Ciddi sonuçları olabilecek gösterge/düğmeler ulaşılması zor bir yerde midir?					13		18	
5	Sistem, kullanıcıları sonuçları ciddi ve yıkıcı olabilecek hatalar yapmak üzere iken onları uyarıyor mu?						10	11	
7. Hatırlama Yerine Tanıma									
6	Anlamlı grupları belirlemek için çerçeveler kullanılmış mıdır?				4				
7	İlişkili elemanları gruplamak için aynı renkler kullanılmış mıdır?				4				
8. Esneklik ve Verimlilik									
1	Eğer sistem acemi ve profesyonel kullanıcıların ikisini de destekliyor ise detaylı hata mesajları farklı düzeylerde sağlanmış mıdır?		10						
9. Estetik ve Sade Tasarım									
6	Sistem, veri girmek için kutu dışında açılır kutu, radyo butonu gibi başka seçenekler sunuyor mu?	10							
10. Yardım ve Dokümantasyon									
1	Çevrim içi bilgilendirmeler (online yardım gibi) görsel ola-rak ayırt edilebilir durumda mıdır?				9				
3	Yardım işlevi görünür mü? Örneğin YARDIM adında bir düğme veya özel bir menü mevcut mudur?			10					
5	Kullanıcı, var olan detay seviyesini değiştirebiliyor mu?			17					
6	Yardım sistemine kolayca ulaşılabiliyor mu veya bu bölümden kolayca çıkılabiliyor mu?				12				
11. Yetenekler									
4	Kullanıcılar, bir alan içerisinde ileri ve geri hareket edebiliyorlar mı?			6	4				
5	Fonksiyonelliği desteklemeye yetecek kadar sayıda, ancak aramayı ve bulmayı zorlaştırmayacak kadar yeterli işlev komutu (ekranda) var mı?			8					
12. Kullanıcı ile Seviyeli Bir İletişim									
1	Her bir ikon (simge), bir ikon grubunun uyumlu bir öğesi midir?					5			
6	Sistem, veri giriş alanlarına bir bölümü girilmiş verileri tamamlıyor mu?							10	
	Toplam	**21**	**111**	**84**	**43**	**60**	**55**	**49**	**0**

SORUNLARIN YOĞUNLAŞTIĞI SORULAR ve SAYILARI		İLİŞKİLİ OLDUĞU KAVRAM			Scratch		Small Basic		Kodu Game		Robomind	
1. Sistem Durumunun Görünürlüğü		Kullanılabilirlik	Tasarım	Her ikiside	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok
2	Her menüdeki bilgilendirmeler, hatırlatmalar ve hata mesajları sistem içerisinde aynı yerde görünmekte midir?			x		10						
3	Eğer hata mesajları açılan bir alan içerisinde gösteriliyor ise, kullanıcı, hatalı olan alanı görebilmekte midir?	x				9						
2. Sistem ve Gerçek Dünyanın Eşleşmesi												
2	Eğer görsel bir ipucu olarak kullanılan şekil varsa, bu kültürel geleneklerle eşleşiyor mu?		x						9			
6	Sistem kullanıcı jargonunu kullanıp ve bilgisayar jargonundan sakınmakta mıdır?	x							12			
3. Kullanıcı Kontrol ve Özgürlüğü												
2	Açılır pencere olan sistemlerde, kullanıcılar için pencereler arası geçiş kolay mıdır?	x				10						
3	Herhangi bir işlem hareketi, veri girişi veya birçok diğer işlem hareketleri için "geri al" fonksiyonu bulunmakta mıdır?			x	11							
4. Tutarlılık ve Standartlar												
5	Menüler dikey olarak sunulmuş mudur?		x						11			
6	Eğer "çıkış" bir menü seçeneği ise listenin her zaman en altında yer almakta mıdır?		x					10	10			
5. Kullanıcıların hataları tanımasına, onları belirlemesine ve önlemesine yardımcı olma												
1	Bir hatayı bildirmek için ses kullanılıyor mu?		x				18					
2	Hatırlatıcılar, işlemin kullanıcının kontrolünde olduğunu ima etmekte midir?	x				9						
3	Hatırlatıcılar, kısa, net ve anlaşılır mı?	x				11						
4	Hata mesajları kullanıcının değil, sistemin sorumlu olduğunu belirtecek şekilde ifade edilmekte midir?		x			13	12			12		
5	Hata mesajları gramer açısından doğru mudur?		x			14						
6	Hata mesajları sorunun nedeni hakkında bilgiler sunuyor mu?		x			13				11		
7	Hata mesajları, kullanıcıya hatayı düzeltmek için hangi hareketi yapması gerektiğini belirtiyor mu?			x		12	13			12		

SORUNLARIN YOĞUNLAŞTIĞI SORULAR ve SAYILARI		İLİŞKİLİ OLDUĞU KAVRAM			Scratch		Small Basic		Kodu Game		Robomind	
6. Hataları Önleme		Kullanılabilirlik	Tasarım	Her ikiside	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok	Hayır	U.Yok
2	Veri girişleri mümkün olduğu kadarıyla büyük ve küçük harfe duyarlı olarak girilebiliyor mu?			x						10	10	
3	Ciddi sonuçları olabilecek gösterge/düğmeler ulaşılması zor bir yerde midir?			x					13		18	
5	Sistem, kullanıcıları sonuçları ciddi ve yıkıcı olabilecek hatalar yapmak üzere iken onları uyarıyor mu?	x								10	11	
7. Hatırlama Yerine Tanıma												
6	Anlamlı grupları belirlemek için çerçeveler kullanılmış mıdır?		x					4				
7	İlişkili elemanları gruplamak için aynı renkler kullanılmış mıdır?		x					4				
8. Esneklik ve Verimlilik												
1	Eğer sistem acemi ve profesyonel kullanıcıların ikisini de destekliyor ise detaylı hata mesajları farklı düzeylerde sağlanmış mıdır?			x		10						
9. Estetik ve Sade Tasarım												
6	Sistem, veri girmek için kutu dışında açılır kutu, radyo butonu gibi başka seçenekler sunuyor mu?		x		10							
10. Yardım ve Dokümantasyon												
1	Çevrim içi bilgilendirmeler (online yardım gibi) görsel ola-rak ayırt edilebilir durumda mıdır?		x					9				
3	Yardım işlevi görünür mü? Örneğin YARDIM adında bir düğme veya özel bir menü mevcut mudur?		x				10					
5	Kullanıcı, var olan detay seviyesini değiştirebiliyor mu?			x			17					
6	Yardım sistemine kolayca ulaşılabiliyor mu veya bu bölümden kolayca çıkılabiliyor mu?			x				12				
11. Yetenekler												
4	Kullanıcılar, bir alan içerisinde ileri ve geri hareket edebiliyorlar mı?			x			6	4				
5	Fonksiyonelliği desteklemeye yetecek kadar sayıda, ancak aramayı ve bulmayı zorlaştırmayacak kadar yeterli işlev komutu (ekranda) var mı?			x			8					
12. Kullanıcı ile Seviyeli Bir İletişim												
1	Her bir ikon (simge), bir ikon grubunun uyumlu bir öğesi midir?			x					5			
6	Sistem, veri giriş alanlarına bir bölümü girilmiş verileri tamamlıyor mu?		x								10	
	Toplam				21	111	84	43	60	55	49	0

Öz Geçmiş

Doğum Yeri ve Yılı	Kertmekaracaveran - 1981		
Öğrenim Gördüğü Kurumlar	**Başlama Yılı**	**Bitirme Yılı**	**Kurum Adı**
Lise	1995	1999	Sivas Teknik ve Endüstri Meslek Lisesi
Lisans	2000	2004	Selçuk Üniversitesi
Yüksek Lisans	2013	2016	Uludağ Üniversitesi
Bildiği Yabancı Diller ve Düzeyi	İngilizce- Orta		
Çalıştığı Kurumlar	**Başlama ve Ayrılma Tarihleri**	**Kurum Adı**	
	2005-	MEB- Bilişim Teknolojileri Öğretmeni	
Yayınlanan Çalışmalar	Baltalı, S. (2011). *Jquery*. (7.Baskı). İstanbul: Kodlab Yayınevi. Baltalı, S. (2013). *Jquery Mobile*. (1.Baskı). İstanbul: Kodlab Yayınevi.		
Mesleki Topluluklar	Bilişim Teknolojileri Eğitimcileri Derneği		

26/04/2016

Salih BALTALI