Elementary Education Online, 13(4), 1531-1541, 2014.
İlköğretim Online, 13(4), 1531-1541, 2014. [Online]: http://ilkogretim-online.org.tr
DOI: 10.17051/io.2014.54872

Factors Effecting Information Technology Teacher Trainees' Programming Process

Fatih ÖZDİNÇ [1], Arif ALTUN [2]

ABSTRACT. The aim of this study is to determine the problems effecting programming process of pre-service information technology teachers and the reasons of these problems by monitoring programming process. In this study, programming processes of pre-service teachers were investigated with cognitive task analysis. Six Information Technologies teacher candidates, taking Internet Based Programming course in Fall Semester of 2011-2012 academic year at the Department of Computer Education and Instructional Technologies (CEIT) at Hacettepe University, participated in the study. Participants were divided to three groups, namely advanced, intermediate and beginner according to their prior programming knowledge and experiences, Tasks of writing program and reading program were assigned to the participants. After the tasks of reading program and thinking and writing program simultaneously, interviews were made with retrospective thinking aloud procedure. The data was analyzed by the NVivo 8 package program. It was found that while participants in advanced level were more successful in the task of writing program, participants in intermediate level were more successful in the task of reading program. The findings indicated that these two processes, writing program and reading program, may require different skills and cognitive processes.

Keywords: Programming, pre-service information technology teachers, cognitive processes

SUMMARY

Purpose and Significance: Positive effects of programming on students are stated in the literature. In order to perform the complex process of programming effectively, pre-service teachers are expected to have programming language at adequate level. The purpose of this study is to determine pre-service IT teachers' programming process and factors effecting programming performance and to determine the relationship between these factors.

Method: This qualitative study was conducted with six pre-service IT teachers at the Department of Computer Education and Instructional Technologies (CEIT) at Hacettepe University in the Fall Semester of 2011-2012. The participants were divided into three groups including advanced, intermediate and beginner based on their previous programming knowledge and experience. Two tasks including writing and reading a code block was given to participants. Cognitive task analysis was conducted to determine the factors effecting the programming process of pre-service information technology teachers. In the task of writing a program, retrospective thinking aloud procedures were applied, whereas in the task of reading the program, a think aloud procedure was conducted.

Results: It was found that while the performances of participants in the advanced level were better in the process of program writing, the participants in the intermediate level are better in the process of reading the program codes. This finding indicated that reading and writing program codes may require different skills and demand various cognitive processes.

Discussion and Conclusion: In the courses of programming languages at the undergraduate level, both writing program codes and reading them as tasks are required and aimed to be developed gradually. The results of this study imply that these two tasks should not be used interchangeably. In other words, these two processes indicate differing skills and processes. When designing instructional tasks and developing course syllabi, course instructors may take these findings into account.

[1] Dr., Hacettepe University, fatihozdinc@gmail.com
[2] Prof. Dr., Hacettepe University, altunar@hacettepe.edu.tr

Bilişim Teknolojileri Öğretmeni Adaylarının Programlama Sürecini Etkileyen Faktörler

Fatih ÖZDİNÇ [1], Arif ALTUN [2]

ÖZ. Bu çalışmanın amacı, bilişim teknolojileri öğretmen adaylarının programlama sürecini izleyerek, bu süreçteki sorunları ve bu sorunların nedenlerini belirlemektir. Bu çalışmada bilişsel görev analizi ile öğretmen adaylarının programlama süreçleri incelenmiştir. Çalışmaya Hacettepe Üniversitesi Bilgisayar ve Öğretim Teknolojileri Eğitimi (BÖTE) Bölümü'nde 2011-2012 Güz Dönemi'nde İnternet Tabanlı Programlama dersine devam eden 6 bilişim teknolojileri öğretmen adayı katılmıştır. Katılımcılar önceki programlama bilgi ve deneyimlerine göre ileri, orta ve düşük olmak üzere üçe ayrılmıştır. Katılımcılara program yazma ve program okuma görevleri verilmiştir. Program okuma görevi ile eş-zamanlı düşünme ve program yazma görevinden sonra geriye dönük sesli düşünme ile görüşme yapılmıştır. Veriler NVivo 8 programı yardımıyla analiz edilmiştir. Program yazma görevinde ileri düzeydeki katılımcılar daha başarılı olurken, program okuma görevinde orta düzeydeki katılımcıların daha başarılı olduğu görülmektedir. Bu durum, program yazma ve okuma görevlerinin birbirinden farklı beceriler gerektirdiğini ve farklı bilişsel süreçler içerdiğini göstermektedir.

Anahtar Kelimeler: Programlama, bilişim teknolojileri öğretmen adayları, bilişsel süreçler

GİRİŞ

Bilişim Teknolojileri dersinin verimliliği, bir dönem seçmeli ders statüsünde oluşu ve ders kapsamında bilgisayarın daha çok okuma olarak ele alındığı, yazma boyutuna gereken önem verilmediği konusu uzun süre tartışılmıştır. Oysaki ilkokul ve ortaokul öğrencilerine programlama ve tasarım araçlarının öğretilmesinin problem çözme, analitik düşünme becerileri, uzamsal düşünme becerileri ve işbirlikli çalışma gibi çeşitli becerilerin gelişmesinde olumlu etkileri bulunmaktadır (Akpınar ve Altun, 2014). Milli Eğitim Bakanlığı, 2013 yılında dersin yazarlık boyutunu genişleterek Bilişim Teknolojileri ve Yazılım dersini ortaokul 5 ve 6. sınıflarda zorunlu ders olarak programa eklemiştir. Böylece, bu dersi tasarlayacak ve programlama öğretecek olan bilişim teknolojileri öğretmenlerinin programlama bilgileri daha önemli hale gelmiştir. Bilişim teknolojileri öğretmenlerinin karmaşık olan bu süreci etkili olarak yürütebilmeleri için programlama konusunda yeterli bilgi birikimine ve donanıma sahip olması beklenmektedir. Bu nedenle öğretmenlerin bu birikimi sağlayacağı lisans eğitimlerindeki programlama süreçlerinin incelenmesi önem kazanmaktadır.

Programlama becerisi bilgisayar bilimi ile ilgili alanlarda önemli bir beceridir ve bu alanda giriş düzeyinde öğrenim gören öğrenciler tarafından programlama dersi zor olarak algılanan bir derstir (Aşkar ve Davenport, 2009). Programlama dili öğrenme sürecinde çoğu işlem ve kavram öğrenciler açısından soyut kalmakta ve öğrenenler öğrendikleri bilgileri somutlaştırmakta zorlanmaktadırlar (Ersoy, Madran ve Gülbahar, 2006). Programlamaya yeni başlayanların öz-yeterlik algıları da düşüktür (Altun ve Mazman, 2012). Bu durum öğrenenlerin programlamadan uzak durmasına ve programlamayı öğrenememesine neden olmaktadır.

Programlama derslerinde performansı etkileyen birçok faktör bulunmaktadır. Bunlardan bazıları motivasyon, programlamaya karşı tutum, programlama dilinin karmaşıklığı ve öğretim yöntem ve tasarımı olarak sıralanabilir (Jenkins, 2002). Bunun yanında programlama performansını etkileyen faktörleri belirlemede birçok model geliştirilmiştir. Wilson (2002) bilgisayar programlama dersinde, programlama performansını etkileyen rahatlık düzeyi, matematik alt yapısı, şansa bağlama, programlamada formal sınıf yapısı ve oyun oynama olmak üzere toplam beş faktör olduğunu belirmektedir.

Programlama performansını etkileyen faktörleri inceleyen çalışmalara bakıldığında, genellikle süreç sonunda yapılan testlerle performans ölçülmüş ve gelişim incelenmiştir. Ancak programlama sürecine yönelik ayrıntılar eksik kalmaktadır. Bireylerin programlama sürecinde bir görevi yerine

[1]Dr., Hacettepe Üniversitesi, fatihozdinc@gmail.com
[2] Prof. Dr., Hacettepe Üniversitesi, altunar@hacettepe.edu.tr

getirme süreçlerindeki bilişsel süreçlerinin ortaya konulması, görevlerin tasarımında bireyleri başarıya ya da başarısızlığa götüren yöntemlere ipucu olması bakımından önemlidir. Dolayısıyla, bu çalışmanın amacı bilişim teknolojileri öğretmen adaylarının programlama süreçlerini izlemek ve programlama performansını etkileyen faktörleri ve bu faktörler arasındaki ilişkileri belirlemektir.

YÖNTEM

Bu çalışmada bilişsel görev analizi kullanılmıştır. Bilişsel görev analizi, gözlemlenebilir görev performansların gerçekleşmesinde etkili bilgi, düşünce süreci ve hedef yapılara ilişkin açıklama ortaya koyan ve planlama, tanılama ve karar verme gibi önemli bilişsel etkinlikleri içeren görev analizi tekniklerinin bir uzantısıdır. (Yusof ve Yin, 2010). Bilişsel görev analizi, insan unsuru (human factors) ve bilişsel psikolojiden yola çıkan nitel bir yöntemdir (Weir ve arkadaşları, 2007).

Çalışma Grubu

Çalışmaya Hacettepe Üniversitesi BÖTE Bölümü'nde 2011-2012 Güz Dönemi'nde İnternet Tabanlı Programlama dersine devam eden 6 bilişim teknolojileri öğretmen adayı katılmıştır. Çalışma grubu amaçlı örneklem modellerinden aşırı veya aykırı durum örneklemesi ile seçilmiştir. Bu örnekleme tekniği derin bir incelemeye tabi tutulabilecek sınırlı sayıda ancak aynı ölçüde de bilgi bakımından zengin durumların çalışılmasını öngörür (Yıldırım ve Şimsek, 2011). Çalışmanın katılımcıları "K1", "K2" gibi kodlanarak belirtilmiştir.

Tablo 1. *Çalışma grubunun özellikleri*

	K1	K2	K3	K4	K5	K6
Deneyim	İleri	İleri	Giriş	Giriş	Giriş	Giriş
Ders Etkinliklerini Gerçekleştirebilme	İleri	İleri	İleri	İleri	Orta	Orta
Cinsiyet	E	K	E	K	K	K

Katılımcıların, uygulama öncesinde 5 hafta boyunca İnternet Tabanlı Programlama laboratuar derslerinde, ders etkinlikleri gerçekleştirme süreçleri gözlenmiş ve programlama deneyimleri konusunda kendileri ile ön görüşme yapılmıştır. Yapılan ön görüşmeler sonrasında gözlemler doğrultusunda Tablo1'deki özelliklere sahip katılımcılar ile çalışılmıştır.

Katılımcılar önceki dönemlerde algoritma ve ön-programlama bilgileri içeren programlama dersi almıştır. Katılımcılardan ikisinin (K1 ve K2) Php konusunda ders öncesinde ön-deneyimleri vardır. Ders sürecinde verilen etkinlikleri yapabilmektedir. Program yazarken farklı stratejiler denemektedir. Diğer 4 katılımcı Php öğrenmeye yeni başlamıştır. Bunlardan ikisi (K3 ve K4) derste öğrendiklerini uygulayabilmektedir. Ancak yeni bir yaklaşım göstermemektedir. Diğer 2 katılımcı (K5 ve K6) ise öğrenmeye yeni başlamış, bazen verilen görevleri yapabilmekte, görevler çeşitlendikçe öğrendikleri kodları kullanmakta zorluklar yaşamaktadırlar.

Görevler, Kullanılan Araçlar ve Veri Toplama Süreci

Programlama görevlerini içeren bilişsel süreç çalışmaları yazılım geliştirme sürecini anlamada çok önemlidir (von Mayrhauser ve Vans, 1996). Düşünme süreçlerini analiz etmek için eş-zamanlı ve geriye dönük sesli düşünme yöntemleri uzun yıllar kullanılmıştır. Program yazımında bilişsel süreçleri inceleyen araştırmalarda, Sloway ve arkadaşları 1980 ve 1990'larda sesli-düşünme yöntemine öncülük etmişlerdir (Bednarik ve Tukiainen, 2006). Sözel ifadelerin analiz edilmesi çeşitli deneyimlere sahip programcıların bilişsel süreçlerine erişimde ve hangi yöntemleri kullandığına dair önemli veriler sağlamaktadır. Sözel ifadelerin geçerliğini, doğrudan gözlem, video kaydı, görüşme protokolleri yükseltmektedir (Ko ve Uttl, 2003).

Bednarik ve Tukiainen (2006), programcıların programlama sürecinde programı anlama, hata bulma süreçlerinde geçirdiği bilişsel süreçlerin belirlenmesinin bu sürece ışık tutulması açısından önemli olduğunu belirtmektedir. Bu çalışmada katılımcıların programlama süreçlerinin izleneceği iki

görev oluşturulmuştur. Görevlerin oluşturulma sürecinde ileri düzey Php bilgisine sahip, program yazan bir uzman ile İnternet Tabanlı Programlama dersini vermekte olan öğretim elemanının görüşleri alınmıştır. Uzman görüşleri doğrultusunda, değişkene değer atama, döngü, koşul yapısı ve ekrana yazdırma fonksiyonlarının görevlerde kullanılmasına karar verilmiştir. Katılımcılar bu görevleri önce yazma ve okuma görevlerine sırasıyla iki hafta içinde farklı günlerde katılmışlardır.

Program yazma görevi

Bu görevde katılımcılardan istenen sonucu ekranda yazdıracak bir program yazmaları istenmiştir. Bu amaçla katılımcılara şu görev verilmiştir:

"Bir diziye 10 tane sayı girerek, bu sayıları 45'ten büyük olma durumuna göre sınayınız. 45'ten büyük ve küçük olan sayıların adedini hesaplatınız. Ekranda 'x tane küçük, y tane yüksek not' şeklinde yazdırınız."

Programın yapısı gereğince, katılımcıların verilen programı yazabilmeleri için dizi, döngü, koşul ifadesi ve ekrana yazdırma komutları kullanması zorunlu tutulmuştur.

Yazdıkları programın sadece bir Php dosyasından oluşması istenmiştir. Katılımcıların, geliştirdikleri programları denemeleri için yerel sunucuda bir dosya oluşturulmuştur. Katılımcıların programlama süreçleri ekran kaydetme yazılımı ile kayda alınmıştır. Katılımcılara görevi tamamlamaları için serbest süre verilmiş ve hazır kod alıp kullanmamaları koşuluyla İnterneti kullanabilmeleri tercihlerine bırakılmıştır. Katılımcılar programlama sürecinde yalnız bırakılmışlardır. Kod yazmayı sonlandırdıkları zaman ekran kaydı da bitirilmiştir. Sonrasında geçirdikleri programlama sürecine dair katılımcılarla geriye dönük sesli düşünme yoluyla görüşme yapılmıştır. Görüşme, ses kayıt cihazı ile kaydedilmiştir.

Program okuma görevi

Katılımcılara ders süreçlerinde karşılaşmadıkları, doğru çalışan bir programın kodları verilmiştir. Program, 1'den 100'e kadar sayıları bir "For" döngüsünde döndürmekte. 10'a bölünen sayıları alt alta ekrana yazdırmaktadır. En sonunda "Toplam=" yazarak karşısına 1 ile 100 arasındaki 10'a bölünen bu sayıların toplamını yazdırmaktadır. Program okuma görevi için verilen kod Şekil 1'de gösterilmektedir.

```
<?php
 $toplam=0;
 for($i=1;$i<=100;$i++)
    {if($i%10==0 )
        {echo $i, "<br>";
            $toplam=$toplam+$i;}
    }
 echo "Toplam=",$toplam;
 ?>
```

Şekil 1. Program okuma görevinde katılımcılardan okunması istenen kod

Program, katılımcılara A4 kâğıt üzerinde siyah-beyaz renkte verilmiştir. Kod, programlamada kullanılan girinti-çıkıntı yapılarına uygun olarak yazılmıştır. Katılımcılardan bu programda aşama hangi işlemlerin yapıldığını ve ne sonuçlar vereceğini hakkında eş-zamanlı olarak sesli düşünmeleri istenmiştir. Program sonunda ekranda görüntülenecek çıktıyı kâğıdın alt kısmına yazmaları istenmiştir. Katılımcıların konuşmaları ses kayıt cihazı ile kaydedilmiştir.

Verilerin Analizi

Verilerin analizine başlamadan önce kod yazma görevinde geriye dönük sesli düşünme ve kod yazma görevindeki eş-zamanlı sesli düşünme her katılımcının görüşme kaydı ayrı doküman olacak şekilde metin haline getirilmiştir. Bu dokumanlar NVivo 8 nitel veri analizi programına aktarılmıştır. Program yazma ve program okuma görevleri için iki ayrı *.nvp dosyasında çalışma yürütülmüştür. Program yazma sürecini içeren metin dosyaları, katılımcıların süreçte ekran görüntüleri ile karşılaştırılmış ve kod şeması oluşturulmuştur. Benzer işlem kod okuma görevinde metin haline

getirilen dokumanlar, katılımcıların sonuçları yazdıkları A4 kâğıtlardaki programlama aşamaları ile karşılaştırılarak kodların doğrulanması sağlanmıştır.

Program yazma görevinde kodlama şeması oluşturulurken katılımcıların program yazma süreçleri ve bu süreçte karşılaştıkları zorluklar temel alışmıştır. Bu sürece ait nitel verilerin analizinde öncelikli olarak 20 serbest kod (free node) oluşturulmuştur. Tüm kodlar ortaya çıktıktan sonra dokümanlar yinelemeli kodlanarak yeni oluşan kodlar tüm dokümanlarda kontrol edilmiştir. Benzer kodlar ortak bir ağaç kod yapısında toplanmıştır (tree node). Böylece katılımcıların program yazma sürecini etkileyen 6 faktör belirlenmiştir.

Program okuma görevinde kodlama şeması oluşturulurken katılımcıların programı nasıl okudukları ve bu süreci etkileyen değişkenler göz önüne alınarak oluşturulmuştur. Program okuma görevine ait dokümanlarda 16 serbest kod belirlenmiştir. Benzer yönü olan kodlar ağaç kodlara dönüştürülmüştür. Kodlamaların bitiminde katılımcıların program okuma sürecini etkileyen 5 ana faktör belirlenmiştir.

Kodlama sürecinde ve sonrasında kodlar arasındaki ilişkiler ve ilişki türleri belirlenmiştir. Sonrasında kodlara ve aralarındaki ilişkiyi göz önüne alan modeller, program yazma ve program okuma süreçleri için ayrı ayrı çıkartılmıştır.

Araştırmalarda verilerin çeşitlenmesi, yanlılığı en aza indirerek çalışmanın geçerliğini arttıracaktır (Cresswell, 2007). Bu araştırmada, program yazma görevinde ekran görüntü videoları, program yazma görevinde dokümanlar ile görüşme kayıtları verileri desteklenmiş ve verilerin çeşitlenmesi sağlanmıştır.

Çalışmada verilerin analizi sürecinde kodlayıcılar arası uyum incelenmiştir. Kategorilerin belirlenmesinde ve indirgenmesi sürecinde programlama konusunda deneyimli bir alan uzmanından yardım alınmıştır. Dokümanlardan alınan örnekler alan uzmanına verilerek, cümleleri kodlaması istenmiştir. Miles ve Huberman'nın önerdiği gibi kodlayıcılar arası uyum yüzdesi hesaplanmıştır. Uyum yüzdesi program yazma görevi için %79, program okuma görevi için %83 bulunmuştur. İki görevin kodlarını birleştirilerek hesaplandığında kodlayıcılar arası uyum yüzdesi %80 olarak hesaplanmaktadır. Bu değerin %70 olması çalışmaların güvenirliklerinin sağlanmasında yeterli görülmektedir (Miles ve Huberman, 1984).

BULGULAR

Program Yazma Görevine Ait Bulgular

Program yazma görevinde 6 katılımcının 4'ü program yazma işlemini tamamlamıştır. Tablo 2'de katılımcıların program görevini tamamlamasına yönelik bilgiler gösterilmektedir.

Tablo 2. *Program yazma görevinin tamamlanması*

	K1	K2	K3	K4	K5	K6
Program Yazma Görevi	+	+	+	-	+	-

+ :Tamamladı -: Tamamlayamadı

K1 katılımcısı, program yazma görevini tamamlamıştır. Kod girinti ve çıkıntılarına özen göstermiş, dizi, döngü ve koşul ifadelerini yerinde kullanarak çalışan bir programla görevi bitirmiştir. K2, istenen göreve uygun bir program yazarak, bu görevi başarı ile tamamlamıştır. Girinti ve çıkıntılara dikkat etmekle birlikte bu konuda K1 katılımcısı kadar başarılı değildir. K3 görevi tamamlamıştır. K4, program yazma görevini tamamlayamamıştır. Yazdığı kodlarda hata yapmamıştır. Ancak sayaç algoritmasını zihninde canlandırmamıştır. Döngü her döndüğünde bir değişkenin kendine 1 eklemesini atlamıştır (ör: $x=$x+1). Bu yüzden farklı bir program yazmıştır. K5, verilen görevi tamamlamıştır. Internet üzerinden ders notlarına ulaşarak programın nasıl yazılacağı konusunda bilgi edinmiştir. K6, program yazma görevini gerçekleştirememiştir. Yazılan kodlarda yalnızca dizi tanımlama kısmı doğrudur. Dizinin elemanlarını çağırmak için döngü kullanamamıştır. Koşul yapısı içerisine değişkenleri kontrol etmeye çalışmıştır. K6 tarafından yazılan kod, çalışan programın kod yapısına çok uzaktadır.

Katılımcıların program yazma süreçlerine genel olarak bakıldığında K1 ve K2 katılımcılarının derste öğrendikleri yöntemlerden farklı yöntemler kullanma yönünde bir çabaları olduğu gözlenmiştir. Diğer katılımcılar genellikle derste öğrendikleri yöntemi uygulamışlardır. Benzer yöntemi önceden derste uygulamayanlar ise bu görevi gerçekleştirmede başarısız olmuştur. Bu durum bilgi düzeyi yüksek olan katılımcıların program yazma sürecinde alternatif yollar denediklerini göstermektedir.

Program yazma sırasındaki yapılan gözlemlerde ve sonradan ekran görüntüleri incelendiğinde, program yazma sürecinde K1, K2 ve K3 katılımcılarının, program yazmaya başladıklarında daha kararlı bir biçimde yazma işlemlerini sürdürdükleri gözlemlenmiştir. K4 ve K6 katılımcıları ise program yazmaya başlamış daha sonra yazdıklarını silerek yeniden yazmaya başlamıştır. Bu katılımcıların programı yazma görevini tamamlayan katılımcılara göre daha kararsız oldukları gözlemlenmiştir. K4 katılımcısı program yazma görevinde yaşadığı kararsızlığı şu şekilde belirtmektedir:

[K4]: *"Yönergede 'sayıları yazdırın' demeden ben sayıları yazdırmaya kalktım. İlk önce sayıları yazdırdım sonra kararsız kaldım, düzeltmeye çalıştım."*

Program yazma sürecinde, bilgi düzeyi daha düşük olan katılımcıların daha çok kararsızlık yaşadığı görülmektedir.

Php'de kod yazmada deneyimi olan katılımcılar program yazmaya başlamadan önce zihinlerin bir tasarlama sürecinin önemli olduğunu belirtmişlerdir. Kod yazmaya başlamadan önce zihinlerinde programın akışını tasarladıklarını belirtmişlerdir. Katılımcılardan birisi (K1) kâğıt üzerinde tasarlamanın program yazma sürecinde daha etkili olacağını belirtmiştir. K3 katılımcısı doğrudan kod yazmaya başladığını belirtmiştir. Kod yazdıkça şekillendirdiklerini belirtmiştir. Deneyimi ve kod bilgisi daha az olan katılımcılardan. K6 ise program yazmaya başlamadan önce kâğıt üzerinde bir tasarlama yapmaya başlamış ancak çözüme ulaşamayınca hemen yazmaya başlamıştır. K6'nın tasarlama sürecinin çok kısa olduğu gözlenmiştir. Ancak K1 katılımcısı kâğıt üzerinde, kodu nasıl yazacağını açıklığa kavuşturana kadar yazmaya başlamamıştır. K5 bir tasarlama yapmadan kaynaklardan yardım aldığını belirtmiştir:

[K5]: *Döngünün kaç defa döndüğünü bulmak için sayaç gibi bir şey mi yapacağım diye İnternete bakmıştım.*

Yazmadan önce tasarlama sürecine genel olarak bakıldığında, katılımcıların yazmadan önce zihinden tasarlama süreci geçirdikleri görülmektedir. Deneyimli katılımcılar zihinlerinde yaptıkları tasarımı kâğıt üzerinde tasarlamakta ve tasarım sonrasında bir sonuç elde ettiği zaman programı yazmaya başlamaktadır. Deneyimi ve bilgisi daha az olan katılımcılardan birinin kâğıt üzerinde tasarlama çabası olmakla birlikte bu tasarımı tamamlayamadan program yazmaya başladığı gözlemlenmiştir.

Program yazma görevinde başarısız olan K4 neden başarısız olduğunun farkındadır ve yeniden yaparsa düzeltebileceğini belirtmektedir:

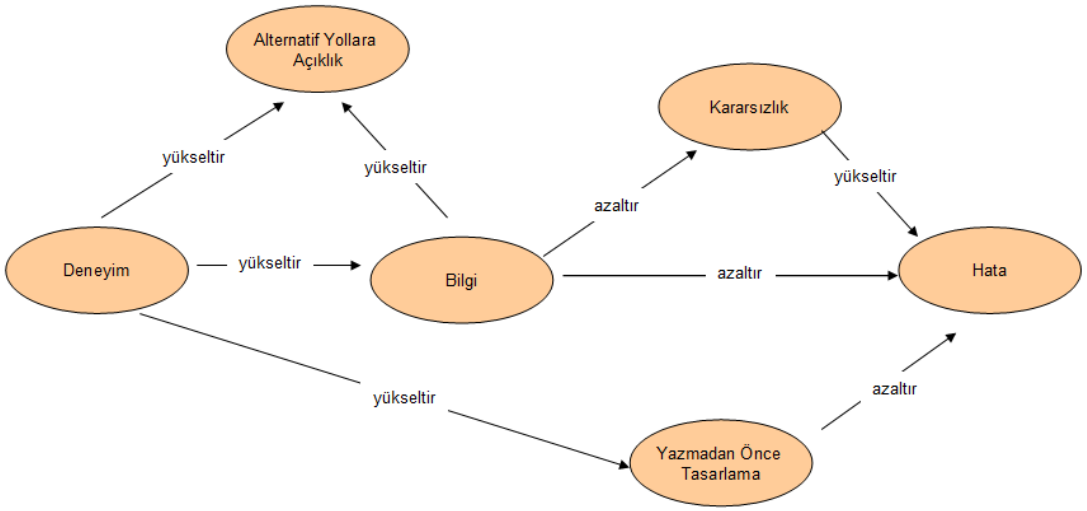
[K4]:*"….döngünün kaç defa döndüğünü yazdıramadım. "For" döngünün içine koyduğum zaman onun kaç defa döndüğüne hesaplattıramadığım için. … for döngüsü tanımlayıp oradaki değişkeni 1 arttırmam gerekiyordu. "*

Görevi gerçekleştiremeyen bir diğer katılımcı K6 ise dizilerde hatası olduğunu belirtmektedir. Ancak asıl hatası döngüden diziyi çağırmak kısmındadır:

[K6]: *"Diziyi tanımlayabiliyordum zaten ama bu çok yönlü olanlara geldiğimde karıştırıyorum. ama dizi mantığı daha oturmamış. bu konuda ben eksik olduğumu biliyorum zaten"*

Program yazma görevinde önceden deneyimi olan kullanıcıların bilgi düzeylerinin de daha yüksek olduğu gözlenmiştir. Ayrıca deneyimli katılımcılar daha az hata yapmıştır.

Katılımcıların program yazma sürecine genel olarak bakıldığında, program okuma görevi verileri analiz edilerek alternatif yollara açıklık, deneyim, kararsızlık, yazmadan önce tasarlama, bilgi ve hata temaları oluşturulmuştur. Alternatif yollara açıklık, deneyim, kararsızlık, yazmadan önce tasarlama, bilgi faktörlerinin program yazmada hata yapma sürecindeki ilişikleri Şekil 2'de görüldüğü gibi modellenmiştir.



Şekil 2. Program yazma sonucunda ortaya çıkan örüntü

Şekil 2'de görüldüğü gibi katılımcılar Php ön-bilgi düzeyi daha yüksek olan katılımcılar daha az hata yapmakta, kod yazarken kararsızlık sürecini daha az yaşamakta ve yazma sürecinde derste öğrenilenin dışında alternatif yolları izlemektedir. Daha deneyimli katılımcılar program yazmaya başlamadan önce sürecin tasarlanmasını önemsemektedir. Bu durum, daha az hata yapılmasını sağlamaktadır. Php bilgisi yüksek olan katılımcılar program yazarken farklı yolları deneme eğilimi göstermektedir.

Program okuma görevine ait bulgular

Program yazma görevinde 6 katılımcının 5'i program okuma görevini tamamlamıştır. Yalnızca K6 görevi tamamlayamamıştır. Görevlerini tamamlayan ileri düzey K1 ve K2 katılımcıları, kod okuma sonunda önce eksik sonuç yazmıştır. Araştırmacının, katılımcıdan program okuma sürecini yeniden yapması istediğinde katılımcılar hatalarını fark etmiş ve düzeltmişlerdir. Tablo 3'te katılımcıların görevleri tamamlama durumları gösterilmektedir.

Tablo 3. *Program okuma görevinin tamamlanması*

	K1	K2	K3	K4	K5	K6
Program Okuma Görevi	Önce eksik-Sonra tamamladı (Hatalı)	Önce eksik-Sonra tamamladı (Hatalı)	Sorunsuz tamamladı	Sorunsuz tamamladı	Sorunsuz tamamladı	Tamamlayamadı

K1, döngü dışındaki bir öğeyi döngü içinde gibi düşünerek aslında program sonunda yazdırılması gereken "Toplam=" yazısını her satırda yazdırmıştır. Sonrasında kararsız kalmıştır. Programı yeniden okuduğunda aynı hatayı farklı bir şekilde tekrarlamıştır. Programı tekrar okuduğu zaman doğru sonuca ulaşmıştır. Ancak her satırın sonuna virgül (,) koymuştur. Virgül işaretinin görevini o anda anlayamamıştır.

[K1]: *"Virgül olmaması gerekiyor burada. Nokta olması gerekir. Virgül hata verir 'Echo'yu' çalıştırmaz bildiğim kadarıyla. Virgül çalıştırıyor muydu yaa? (kendi kendine sorar.) Şimdi her şeyi döndürecek ama şu virgül ne olur bilmiyorum ama bildiğim kadarıyla hata verir"*

Bu durumun dikkatsizlik sonucu mu olduğu düşünülmekle birlikte bunun bir yapıya yönelik bilgi eksikliğinden kaynaklandığı görülmektedir. Kod içinde virgül kullanım amacının alternatifi, katılımcı tarafından bilinmekte ancak virgülün görevi bilinmemektedir.

K2, program okuma sürecinde döngü içinde yapılan işleme odaklanmıştır. Dolayısıyla ekranda yalnızca sondaki ekrana yazdırarak toplama sonucunu vermiştir. Sonrasında kodu yeniden okuduğu

zaman döngünün içindeki ekrana yazdırma kodunu görmüş ekran çıktısını düzeltmiştir. Döngünün son dönüşünde aradaki Php kodları arasındaki Html etiketlerinin tekrar çalıştıracağını atlamış ve toplam değerini en alt satıra yazdırmak yerine son rakamın olduğu sayının yanına yazdırmıştır.

[K2]: *"Direk sonuca odaklandım ben. Burada matematiksel işlem var ya. Neyi tutuyor diye baktım. O yüzden 'echoyu' atladım."*

K2'nin bu hatası ekrana yazdırma komutuna göre daha karmaşık bir yapı olan döngüye odaklanmasına, döngü içindeki ayrıntıları düşünmesine ve daha yüzeysel işlemleri yapamamasına neden olmuştur.

K1 ve K2'nin program okuma sürecinde yaptıkları hata türleri birbirinden farklılık göstermekle birlikte bu katılımcıların program okuma görevinde benzer süreçler geçirdiği görülmektedir. K1 ve K2 katılımcıları kodu başta hatalı yorumlamış, sonra düzeltmişlerdir. Bu katılımcıların döngü yapılarını adım adım ve dikkatli bir şekilde yorumladıkları gözlenmiştir. Her iki katılımcı, döngüde daha çok döndürülen toplama işlemine odaklanmıştır. Bu işlemi gerçekleştirirken ekrana yazdırma komutunu gözden kaçırmış ya da sondaki ekrana yazdırma komutunu döngü içerisine almıştır. Bu durum ekrana bir defa yazdırılması istenen "Toplam" ifadesinin her satırda yazdırılmasına sebep olmuştur. K1 ve K2 katılımcıların program okuma sürecinde ayrıntılara odaklanmakta ve hata yapmaktadır.

K1 ve K2 katılımcılarının program okuma görevinde, programı yeniden okuduklarında hatalarının bir bölümünün farkına vardıkları ve düzelttikleri görülmüştür.

[K2]: *" 'Toplam =550' yazar…İlk önce aldığı rakamları 10'u 20'yi aldığı rakamları birer satır atlayarak yazıyor. Ya nasıl atladım bunu!"*

[K1]: *'Toplam=10' yazar. Daha sonra 11, 12 filan girmeyecek 20inci sayı girecek, 20 yazacak. Burayı da kaçırmışız. Daha sonra Toplam=30 yazacak bir altına böyle böyle gidecek. …Ben buldum şimdi olayı. Burada "toplam"ı yazdırmıyor hiçbir şekilde. En sonunda yazacak"*

K6 katılımcısı geriye dönük düşünme sonrasında hatalarının farkına varamamıştır. Bu bulgu, katılımcıların program okuma görevinde bilgi düzeyi ile hatalarının farkına varma arasında bir ilişki göstermektedir.

K3, K4 ve K5 katılımcıları programı doğru şekilde okumuştur. Programda çıkacak sonucu net olarak ortaya koymuşlardır.

[K3… *"for" içerisine tanımlanmış ve $i'in modu 10'a bölündüğünde 0'a eşit olduğu zaman "echo" $i'i yazdıracak. daha sonra $toplamı da $i kadar arttıracak. Başta birinci if döngüsüne bakacak. 10 a bölünen sayıları bulacak. …'Toplam=550' yazacak"*

K6 katılımcısı program okuma görevini tamamlayamamıştır. Bilgi eksikliği nedeniyle döngü içerisinde kaybolmuştur ve 10'un katları yerine 30'un katlarının yazdırılacağını belirtmiştir.

[*K6]: "1 verdiğimde toplamı zaten 0 ile başlamış. 0, 0 1 dediğimde... Direk toplam 1 , 2 diye girecek. Buldum. …İyice kafam karıştı. …30 var, 60 var, 90 var bu kadar başka da yok. Bunları toplayacak. 180 yazdıracak"*

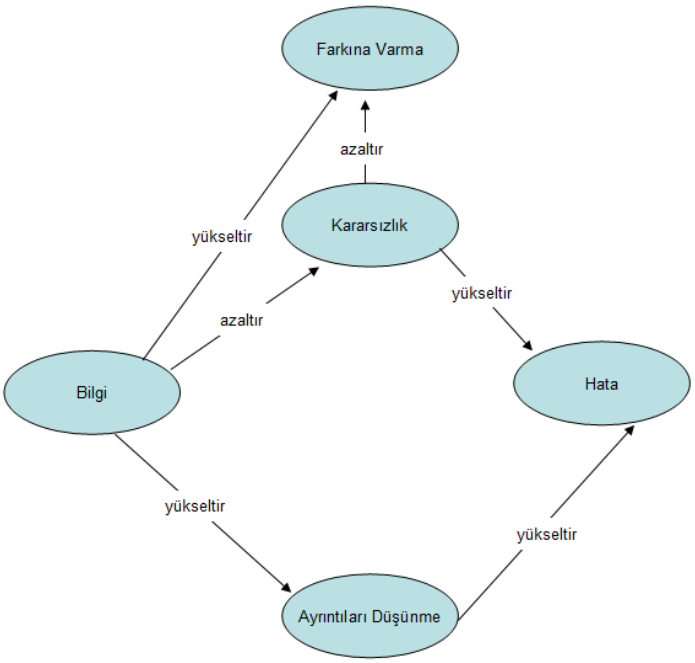
Program yazma görevini başarılı şekilde tamamlayan K1 ve K2 katılımcıları, program okuma görevinde kimi zaman kararsızlığa düşmüştür. Ancak bu kararsız durumlarından hatalarının farkına vardıklarında kurtuldukları gözlemlenmiştir.

[K1]: "*toplam "For" un içinde yazıyor "İf" in içinde değil. Ama.... Bu program çalışmaz. "For" döngüsü kapatılmamış ki... Bu program çalışmaz bence. Şimdi ona dikkat ediyorum "İf "kapatılmış "for" kapatılmamış. hata verir. Hayır konulmuş. Çalışır."*

Düşük bilgi düzeyine sahip katılımcıların daha büyük bir kararsızlık içinde olduğu görülmektedir.

[K6]: *"1 verdiğimde toplamı zaten 0 ile başlamış. 0, 0 1 dediğimde... Direk toplam 1 , 2 diye girecek …. İyice kafam karıştı"*

Katılımcıların program okuma görevi verileri analiz edilerek farkına varma, kararsızlık, ayrıntıları düşünme, bilgi ve hata temaları oluşturulmuştur. Farkına varma, kararsızlık, ayrıntıları düşünme, bilgi faktörlerinin program okumada hata yapma sürecindeki ilişikleri Şekil 3'te görüldüğü gibi modellenmiştir.

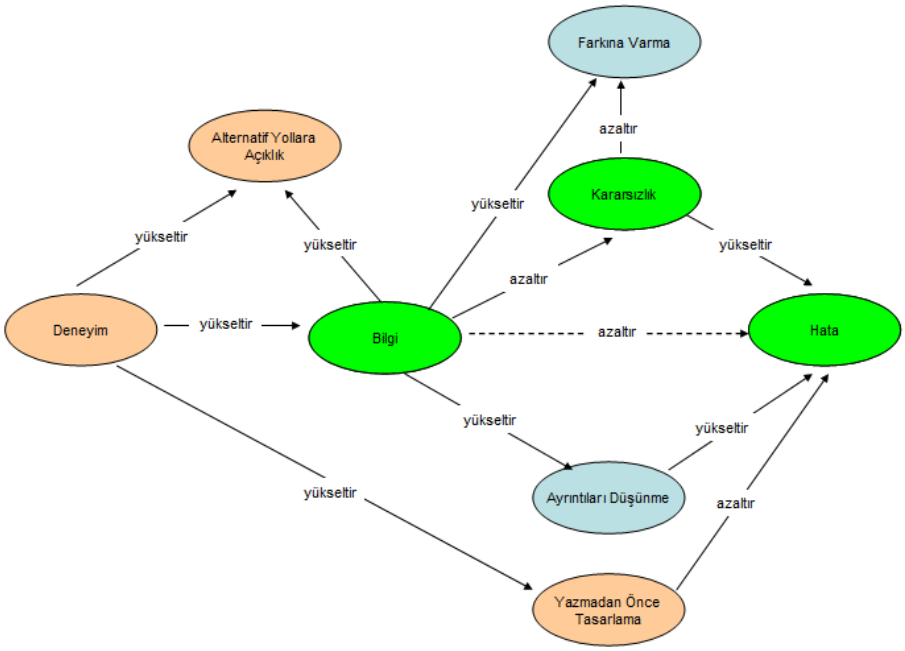


Şekil 3. Program okuma sonucunda ortaya çıkan örüntü

Program okuma görevi bulgularına genel olarak bakıldığında bilgi düzeyi yüksek olan katılımcılar, döngünün içinde ayrıntıları düşünmekte, Kod yapısında ayrıntıları düşünme, katılımcıların sözdizimi konusunda hatalar yapmasına ve hatalı okumalarına neden olmaktadır. İleri düzey katılımcılar, kodu yeniden okudukları zaman hatalarını farkına varmaktadır. Kararsızlığa düşün katılımcılar daha çok hata yapma eğilimi göstermekte, kararsızlık düzeyleri yükseldikçe hatalarının farkına varma olasılıklarının azaldığı görülmektedir.

Program okuma ve program yazma görevlerine ait bulgularının birleştirilmesi

Program yazma ve program okuma sürecindeki ortaya çıkan modeller Şekil 4'te birleştirilerek program okuma ve yazma süreçleri için ortak bir model oluşturulmuştur.



Şekil 4. Program yazma ve okuma görevleri sonucunda ortaya çıkan ortak örüntü

Şekil 4'te görüldüğü gibi bilgi, kararsızlık ve hata öğeleri, program yazma ve okuma süreçlerinde ortak olarak bulunmaktadır. Ortak bulunan bu öğelerde bilgi kararsızlığı azaltırken, kararsızlığın hata olasılığını yükselttiği görülmektedir. Program yazma görevinde bilginin hatayı azaltması, program okuma görevinde ise bilgi ile hata arasında doğrudan bir ilişki bulunmayışı bu ortak modelde iki görev açısından ortak olmayan bir ilişki olarak karşımıza çıkmaktadır. Deneyim, alternatif yollara açıklık ve yazmadan önce tasarlama program yazma sürecine özgü faktörlerdir. Farkına varma ve ayrıntıları düşünme ise program okuma görevine özgü faktörlerdir.

SONUÇ, TARTIŞMA VE ÖNERİLER

Çalışmanın bulguları genel olarak incelendiğinde, program yazma sürecinin ve okuma sürecinin öncelikli olarak bilgi temelinde olduğu görünmektedir. Bilgi sahibi olunsa dahi program okuma görevinde ayrıntıları düşünme, dikkatsizlik, hata yapmaya neden olmaktadır. Program yazma sürecinde ise ön tasarlama, program kodunu yazmak için önemli bir faktördür.

Program yazma ve okumayı etkiyen faktörler bütün olarak ele alındığında bilgi düzeyinin yükselmesinin kararsızlığı azalttığı görülmektedir. Bu konuda deneyimin bilgi üzerindeki etkisinin, programlama sürecinde karar vermede etkili olduğu düşünülmektedir.

Program yazma sürecinde ileri düzeydeki katılımcıların performansları daha üstün iken, program okuma sürecinde daha düşük düzeyde katılımcıların görevi gerçekleştirmede daha başarılı olduğu görülmektedir. Bu durum, program yazma ve okuma performanslarının farklı beceriler ve bilişsel süreçler gerektirdiğini gösterilebilir. Araştırma sonuçlarında iki performansı etkileyen farklı faktörler bulunması bu düşünceyi desteklemektedir.

İleri düzey katılımcılar program yazmaya başlamadan önce diğer gruplara göre daha uzun süre zihinlerinde tasarlama süreci geçirdiğini belirtmektedir. Benzer şekilde, Bednarik ve Tukiainen (2006), program yazma sürecini göz izleme ile inceledikleri çalışmalarında deneyimli katılımcıların önce hipotezler kurduğunu belirtirken, acemi kullanıcıların doğrudan kod yazma sürecine geçtikleri sonucuna ulaşmıştır.

Program yazma görevinde "bilgi hatayı azaltır" ilişkisi doğrudan görülebilirken, program okuma görevinde bu ilişki doğrudan görülememekte hatta belli bir düzeyde bilginin hatayı dolaylı olarak yükselttiği görülmektedir. Bunun sebebi ileri düzey katılımcıların program okurken ayrıntılı düşünerek daha karmaşık yapılara odaklaşarak daha yüzeysel kodları göremeyişidir.

Bu araştırmada deneyimli katılımcıların bilgi düzeyinin daha yüksek olduğu görülmektedir. Döngüyü okuma süreçlerinde ileri düzey katılımcılar döngüyü adım adım döndürmekte, düşük düzey katılımcılar ise yüzeysel olarak incelemektedir. Benzer biçimde, Crosby ve Stelovsky (1990), deneyimin algoritma okumada etkisini araştırdıkları çalışmalarında deneyimin, okuma stratejilerini etkilediğini ve daha deneyimli kişilerin önemli bölgelerde daha fazla zaman geçirdiğini belirtmektedir.

Bu araştırmadan elde edilen sonuçlar bağlamında öğretmen adaylarının programlama sürecini geliştirmeye ve incelemeye yönelik öneriler getirilebilir. Yükseköğretim kurumlarında verilen programlama derslerinde, yalnızca program yazma görevleri yerine program okuma görevlerine de yer verilmelidir. Program okuma ve program yazma performanslarını belirlemeye yönelik yapılan sınavların benzer sonuç vereceği düşünülerek birbiri yerine kullanılması yanlış değerlendirmeye sebebiyet verebileceğinden, programlama derslerinin değerlendirilmesinde bu durum göz önüne alınmalıdır. Programlama öğretimi sürecinin başlarında, programlama öğrenmeye yeni başlayanlara yalnızca program okuma ya da program yazma görevi yerine eksik kodlar verilerek bunların düzeltilmesi uygun bir yöntem olacaktır (Ko ve Uttl, 2003).

Bu çalışmadaki uygulamalar bir dizi, bir döngü ve bir koşul yapısı içermektedir. Program hacmi büyütülerek yeni çalışmalar yapılmalıdır. Oturum, veritabanı kullanımı gibi uygulamaların kullanıldığı daha uzun süreli çalışmalar yapılarak daha karmaşık yapılardaki bilişsel süreçler ortaya konulabilir.

Önceden deneyimi olan katılımcıların program okuma görevinde, döngü ve daha karmaşık kod yapılarına odaklandıkları ve söz dizimi hatalarını gözden kaçırdıkları görülmüştür. Kod yazarken karmaşık yapılara odaklanıldığında söz dizimi hatalarını görebilmek için Php programına özgü editörlerin kullanılması uygun olacaktır.

İleriki çalışmalarda daha çok katılımcıyla çalışılarak, katılımcılardan benzer süreçleri (program yazma ve okuma) gerçekleştirmeleri ve bu süreçte düşündüklerini raporlamaları istenebilir. Katılımcıların dönem başı ön bilgi testleri ve deneyim raporlarıyla bu araştırmada belirlenen faktörlerin sınanması ve faktörlerin daha açık ifade edilmesi sürece katkı sağlayacaktır. Ayrıca program okuma ve yazma görevlerinde göz izleme aracından elde edilen verilerin kullanıldığı araştırmaların yapılması bu araştırmanın sonuçlarını geliştirmeye yardımcı olacaktır.

KAYNAKLAR

Akpınar, Y. ve Altun, A. (2014). Bilgi toplumu okullarında programlama eğitimi gereksinimi. *İlköğretim Online. 13(*1).1-4

Altun, A. ve Mazman, S.G. (2012). Programlamaya İlişkin Öz Yeterlilik Algısı Ölçeğinin Türkçe Formumun Geçerlilik ve Güvenirlik Çalışması. *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 3*(2), 297- 308

Aşkar, P. & Davenport, D. (2009). An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students, *The Turkish Online Journal of Educational Technology – TOJET January. 8*(1).

Bednarik, R. & Tukiainen, M. (2004). Visual Attention and Representation Switching in Java Program Debugging: A Study Using Eye Movement Tracking. In Proceedings of 16th Annual Psychology of Programming Interest Group Workshop (PPIG'04), Institute of Technology Carlow, Ireland, April 5-7, 2004, pp. 159-169.

Bednarik, R., & Tukiainen, M. (2006). An Eye-tracking Methodology for Characterizing Program Comprehension. *In Proceedings the 2006 Symposium on Eye Tracking Research and Applications, ETRA 2006, March 27-29, San Diego, CA, USA,* ACM Press, pp. 125 - 132.

Creswell, J. W. (2007). Educational research: Planning, condacting, and evaluating quantitative and qualitative research (2nd ed*.). Upper Saddle River.* New Jersey: Pearson Education, Inc.

*Crosby, M. E. & Stelovsky, J. (1990) How do we read algorithms? A case study. IEEE Computer, (23)*1, 24–35.

Ersoy, H., Madran, R.O. ve Gülbahar, Y. (2006). Programlama Dilleri Öğretimine Bir Model Önerisi: *Robot Programlama. Akademik Bilişim '07 Konferansı, Kütahya*

Jenkins, T. (2002). On the difficulty of learning to program. *In 3rd annual Conference of LTSN-ICS, Loughbrorough University, Leicestershire, UK*

Ko, A. J., & Uttl, B. (2003). Individual differences in program comprehension strategies in unfamiliar programming systems. *Iwpc 2003: 11th Ieee International Workshop on Program Comprehension*, 175-184.

Miles, M. B., & A. M. Huberman. *Qualitative Data Analysis: A Sourcebook of New Methods.* Beverly Hills, Calif.: Sage, 1984.

von Mayrhauser, a. & Vans, A.M. (1996). Identification of dynamic comprehension processes during large scale maintenance. *IEEE Transactionson Software Engineering 22*,6,424–437.

Weir CR, Nebeker JJ, Hicken BL, Campo R, Drews, F. & Lebar B. A. (2007) . Cognitive task analysis of information management strategies in a computerized provider order entry environment. *J Am Med Inform Assoc 2007;14(*1):65-75.

Wilson, B. C. (2002). A study of factors promoting success in computer science including gender differences. *Computer Science Education, 12*(1–2), 141–164.

Yıldırım, A., ve Şimşek, H., (2011). *Sosyal Bilimlerde Nitel Araştırma Yöntemleri (8. Baskı*). Ankara: Seçkin Yayınevi.

Yusof, N. M. & Yin, W. C. (2010). "Multimedia Learning System (MMLS): Valuing the Significance of Cognitive Task Technique and User Interface Design". *Paper presented in Information Technology (ITSim), 2010 International Symposium. 17 June 2010 Kuala Lumpur, Malaysia. 1 – 6.*