Ankara Üniversitesi Eğitim Bilimleri Fakültesi Dergisi
Yıl: 2018, Cilt: 51, Sayı: 1, 1-26
DOI:10.30964/auebfd.404978, E-ISSN: 2458-8342, P-ISSN: 1301-8342

Yeni Piagetçi Kurama Göre Bilişim Teknolojileri Öğretmen Adaylarının Soyutlama Düzeylerinin ve Programlama Davranışlarının Belirlenmesi

MAKALE TÜRÜ	Başvuru Tarihi	Kabul Tarihi	Erken Görünüm Tarihi
Araştırma Makalesi	23.11.2017	13.03.2018	14.03.2018

Hüseyin Özçınar [1]
Pamukkale Üniversitesi

Öz

Bu araştırmanın amacı bilişim teknolojileri öğretmen adaylarının yeni Piagetçi kurama göre soyutlama becerilerinin belirlenmesidir. Soyutlama düzeylerine göre katılımcıların programlama davranışlarının belirlenmesi bu araştırmanın bir diğer amacıdır. Araştırma nitel araştırma olarak desenlenmiştir. Araştırmanın katılımcıları Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümünde İnternet Tabanlı Programlama dersini alan ve araştırmaya katılmayı kabul eden 15 bilişim teknolojileri öğretmen adayından oluşmuştur. Öğretmen adaylarına iki tane kod okuma bir tane kod yazma sorusundan oluşan toplam üç soru yöneltilmiş, sesli düşünerek soruları yanıtlamaları istenmiştir. Kod okuma soruları kağıt üzerinde, kod yazma sorusu ise bilgisayar ortamında yanıtlanmıştır. Temel veri kaynağı olarak öğrencilerin sesli düşünme kayıtları, yardımcı veri kaynakları olarak da öğrencilerin kod yazma süreçlerine ilişkin video kayıtları ve kod okuma sorularına verdikleri yazılı yanıtlar kullanılmıştır. Araştırmaya katılan öğrencilerden ikisinin yeni Piagetçi kurama göre duyusal-motor düzeyde, altısının işlem öncesi, beşinin ise somut işlemler döneminde olduğu görülmüştür.

Anahtar sözcükler: Yeni Piagetçi kuram, bilgisayar programlama, soyutlama, programlama davranışları

[1] *Sorumlu Yazar*: Yrd. Doç. Dr, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, E-posta: hozcinar@pau.edu.tr, https://orcid.org/0000-0001-8715-2653

Bilgisayar bilimci gibi düşünme ya da hesaplamalı düşünme dünyada birçok ülkede olduğu gibi Türkiye'de de ilköğretim programında yer bulmaya başlamıştır. Bilgisayar programlama hesaplamalı düşünmenin öğretimi için bir yöntem sunmaktadır ve çoklukla hesaplamalı düşünme için bir bağlam olarak kullanılmaktadır. İlköğretim düzeyinde hesaplamalı düşünme dolayısıyla programlama öğretiminin gerekliliği fikri bilgisayar bilimleri ve programlama alanlarında nitelikli bilişim teknolojileri öğretmenlerinin yetiştirilmesini zorunlu kılmaktadır. Ancak bilgisayar bilimleri eğitimi alanyazınında üzerinde neredeyse uzlaşmaya varılan sayılı fikirlerden biri programlama derslerinin öğrenciler tarafından zor bulunduğudur (Corney, Teague, Ahadi ve Lister, 2012; Hanks, 2008). Programlama öğrencileri çoğunlukla eğitimcilerin beklentileriyle orantılı bir gelişim göstermemektedir (McCracken vd., 2001). Bu durum eğitimcinin beklentilerinin öğrencinin düzeyinin üstünde olmasına, öğrencinin bu beklentileri karşılayamamasına dolayısıyla dersin zor ve sıkıcı bir ders olarak algılanmasına neden olmaktadır (Teague ve Lister, 2014).

Soyutlama becerisi birçok eğitimci tarafından bilgisayar bilimleri öğrenmek için gerekli temel beceri olarak nitelendirilmektedir (Bwennedsen ve Caspersen, 2008; Hazzan, 1999; Turner, 1991). Kramer'e (2007) göre iyi bilgisayar programcılarıyla diğerleri arasındaki temel fark soyutlama becerilerinden kaynaklanmaktadır. İyi bir programcının, soyutlama becerisi olarak Piaget'nin soyut düşünme olarak tanımladığı düzeye erişmiş olması gerekmektedir. Ancak yetişkinlerin küçük bir kısmı bu düzeyde mantık yürütebilmektedir (Kramer, 2007). Bu yaklaşımın doğal sonucu olarak Piaget'nin gelişim evrelerinde daha ilerde olan kişilerin programlamada daha başarılı olmaları beklenebilir. Ancak araştırma bulguları bu iddiayı tam olarak desteklememektedir (Lister, 2011).

Piaget'nin bilişsel gelişim kuramına göre bireyler beynin biyolojik gelişimine paralel olarak soyut mantık yürütme becerilerini geliştirirler. Dolayısıyla bir konuda soyut mantık yürütebilen bir bireyin diğer konularda ya da alanlarda da benzer biçimde soyut düşünebileceği öngörülür. Ancak yapılan araştırmalar göstermektedir ki bireylerin soyut düşünme becerileri problem alanına göre değişebilmektedir (Gelman, 1978; Lister, 2011). Piaget'nin bilişsel gelişim kuramına getirilen bu ve benzer eleştiriler yeni Piatgetçi (neo-Piagetian) kuramların ortaya çıkmasını sağlamıştır (Demetriou, 1988). Yeni Piagetçi kuramlar, klasik kuramdan farklı olarak bireylerin soyutlama becerilerinin yaşlarından bağımsız olarak problem alanındaki deneyimleriyle geliştiğini öne sürmektedirler. Bu kuramlara göre bir alanda yeni çalışmaya başlayan birinin mantık yürütme biçimi aynı alanda uzman olan kişiye göre daha az soyut olacaktır (Lister, 2011). Bu bağlamda Lister (2011) ve Teague, Corney, Ahadi ve Lister (2013) yeni Piagetçi kuram çerçevesinde programlama öğrenen bireylerin soyut düşünme becerilerini dört düzeyde incelemiştir.

Duyusal Motor Dönem

Bu dönemde öğrenci yazılı kodları % 50'den fazla doğruluk ile takip edemez. Kavramlar hakkındaki bilgisi bütüncül değildir. Kavramın kullanılacağı yeri doğru

olarak seçemez ya da kavram ya da kod bloğunu doğru biçimde kullanamaz (Perkins ve Martin, 1986; Teague, 2015). Bu dönemdeki öğrenci tarafından problem çözme ve kod yazma sürecinde izlenen temel strateji deneme yanılmadır. Ancak öğrencide kod yazma becerisinin gelişmiş olması beklenmez (Teague, Corney, Ahadi ve Lister, 2013).

İşlem Öncesi Dönem

İşlem öncesi dönemdeki programlama öğrencileri yazılı kodu takip edebilir ve kod içerisindeki değişkenlerin programın akışına göre hangi değerleri alacağını öngörebilir. Ancak kodun genel olarak işlevi ile ilgili bir çıkarımda bulunmakta zorlanırlar. İşlem öncesi dönemdeki öğrenciler herhangi bir anda sadece bir soyut özelliğe odaklanabilirler ve farklı satırlarını birbirleriyle ilişkilendirmekte zorlanırlar. Birden fazla soyutlamada bulunduklarında bu soyut düşünceler arasındaki ilişki karmaşık hatta çelişkili bir biçime dönüşebilir (Gluga, Kay, Lister ve Teague, 2012).

Bu dönemdeki öğrenciler temel düzeyde de olsa kod yazabilirler ancak akış diagramlarını programlama sürecinde kullanmakta zorlanırlar (Teague, 2015). Problem çözme ve kod yazma sürecinde yarı rastlantısal deneme yanılma yöntemini kullanma eğilimi bu dönemin karakteristik özelliklerinden bir diğeridir. Bu yöntemde öğrenci çalışmayan kod üzerinde belirli değişiklikler yaparak sonuca ulaşmaya çalışır ancak bu değişikliklerin anlamı konusunda kesin bir fikre sahip değildir (Teague, 2015).

Somut İşlemler Dönemi

Bu dönemde birey kod ile ilgili soyut mantık yürütebilir ve soyutlamalarda bulunabilir. Ancak bu soyutlamalar hipotetik durumlar için gerçekleştirilemez. Bireyin soyutlamaları daha çok tanıdık ve gerçek durumlar içindir. Somut işlemler dönemindeki öğrenciler inceledikleri kodun amacı konusunda fikir yürütebilirler ve tümden gelimci olarak düşünebilirler (Gluga, Kay, Lister ve Teague, 2012).

Bu dönemdeki programlama öğrencisi ayrıntıların net olarak belirtildiği yönergelerden yola çıkarak küçük çaplı programlar geliştirebilir ancak sadece görev tanımı verildiğinde ya da büyük çaplı bir yazılım oluşturması istendiğinde zorlanır. Bu durumdaki öğrenci soyutlama düzeyini azaltmak için problemin bir alt kümesini, benzer örnekleri inceleyerek çözme eğiliminde olur (Gluga, Kay, Lister ve Teague, 2012). İşlem öncesi dönemdeki programlama öğrencisi sistematik ve üst düzey yazılım planları gerçekleştiremese de üstünkörü planlamalar ve sınırlı işlevsellikte düzeltmeler yapabilir.

Soyut İşlemler Dönemi

Soyut işlemler dönemindeki programlama öğrencisi sistematik ve mantıksal uslamlama gerçekleştirebilir. Karşılaşmadığı hipotetik durumlarla ilgili mantık yürütebilir. Bu dönemdeki öğrencinin kesin bilinen ile olasılıklı durumlar hakkındaki farkındalığı gelişmiştir; eksik veri ile yola çıkarak çözüm geliştirip daha sonra bu çözümü yeni veriler ile sınayabilir (Gluga, Kay, Lister ve Teague, 2012). Bu

düzeydeki öğrenciler bilgilerini daha önce karşılaşmadıkları problemlerin çözümünde yaratıcı bir biçimde kullanabilirler (Teague, 2015).

Araştırmanın Amacı

Öğrencilerin bir problem alanında soyutlama becerisi edinmeleri sınıf içerisinde her öğrenci için aynı hızla gerçekleşmemektedir. Aynı sınıf içerisinde bazı öğrenciler işlem öncesi dönemdeyken bazıları somut işlemler ya da soyut işlemler döneminde olabilmektedir. Bu durumda sınıftaki her öğrencinin, öğretmenin beklentileri doğrultusunda kod yazmasını beklemek programlama derslerinin başarısını düşürebilir. Öğrencilerin yeni Piagetçi gelişim dönemlerinin belirlenerek bu dönemlere uygun yaklaşımlarla soyutlama becerisi kazanmalarının sağlanması daha başarılı bir yaklaşım olacaktır (Lister, 2011). Bu bağlamda bu araştırmanın amacı bilişim teknolojileri öğretmeni adaylarının yeni Piagetçi kurama göre soyutlama becerilerinin belirlenmesidir. Belirlenen soyutlama becerisi düzeylerine göre katılımcıların programlama davranışlarının belirlenmesi bu araştırmanın bir diğer amacıdır.

Yöntem

Bu başlık altında araştırma modeli, sesli düşünme yöntemi, katılımcılar, veri toplama ve verilerin analizi ele alınmıştır.

Araştırma Modeli

Bu araştırma nitel araştırma olarak desenlenmiştir. Nitel araştırmalar en temel anlamda bireylerin ya da grupların problemlere yükledikleri anlamı açıklamak ve anlamak için kullanılan bir yaklaşımdır (Creswell, 2013). Bu çalışmada araştırma amaçlarına ulaşabilmek için öğrencilerin problem çözme süreçleri eş zamanlı sesli düşünme yöntemi ile kayıt edilmiş, bu veriler ekran kayıtları ve problem çözümlerinin gerçekleştirildiği yanıt kağıtları ile birlikte incelenmiştir. Sözlü protokol analizi ya da sesli düşünme yöntemi bilgisayar bilimleri eğitimi alanında farklı araştırma amaçları için birçok kez kullanılmıştır (Özer Şenal ve Erdem, 2017; Teague, Corney, Ahadi, ve Lister, 2013). Bu çalışmada hem öğrencilerin sorulara verdikleri yanıtlar hem de sesli düşünme protokol analizi sonuçları kullanılarak bulguların geçerliliği sağlanmaya çalışılmıştır.

Sesli Düşünme Yöntemi

Sesli düşünme yöntemi bireyin bir görev sürecinde neler düşündüğünün yansız olarak anlaşılabilmesi açısından önemli bir yöntemdir. Bu yöntemle problem çözme sürecindeki bireyin bilişsel süreçleri ve zihinsel modelleri ile ilgili önemli bilgilere ulaşılabilir (Leighton, 2017). Dolayısıyla, sesli düşünme yöntemi bireyin problem uzayını nasıl algıladığı ve hangi bilişsel süreçleri kullanarak çözüm ürettiği ile ilgili önemli bilgiler sağlamaktadır (Hughes ve Parkes, 2003).

Yeni Piagetçi kuramcılar soyutlama becerisinin gelişimini çalışan bellek kapasitesinin verimli bir biçimde kullanılabilmesine bağlamaktadırlar. Çalışan bellek

oldukça sınırlı kapasiteye sahiptir. Ancak gruplama yöntemi ile birbiriyle ilişkili ve uzun dönem hafızada kayıtlı verilerin kısa dönem belleğe tek bir veri gibi çağrılıp işlenmesi mümkün olmaktadır. Gruplama yöntemini etkili bir biçimde kullanabilmesi bir problem alanındaki deneyimin artması ile mümkün olmaktadır ve soyutlama becerisinin gelişimini açıklayan temel değişken olarak kabul edilmektedir (Morra, Gobbo, Marini ve Sheese, 2012). Bireyin problem çözme sürecinde sesli bir biçimde düşünmesinin çalışan belleğinin içeriğini yansıtacağı düşünülmektedir (Ericsson ve Simon 1993). Dolayısıyla sesli düşünme yöntemi programlama öğrencilerinin soyutlama becerilerinin incelenmesi için önemli bir araç sağlamaktadır.

Katılımcılar

Bu araştırmanın katılımcıları Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümünde İnternet Tabanlı Programlama dersini alan ve araştırmaya katılmayı kabul eden 15 bilişim teknolojileri öğretmen adayından oluşmuştur. Ancak katılımcılardan ikisine ait sesli düşünme kayıtları anlaşılır olmadığı için analizler 13 katılımcının verileri üzerinden yürütülmüştür. Katılımcılar öncelikle sesli düşünme yöntemi ile ilgili olarak bilgilendirilmişlerdir. Bunun dışında her katılımcı sesli düşünme ile problem çözme oturumları öncesinde deneme yapmışlardır. Katılımcıların tamamı bu dersi almadan önce eğitim fakültesinde iki programlama dersi almışlardır. Bunun dışında, katılımcılardan meslek lisesinden mezun olan dört kişi lise eğitimleri boyunca iki programlama dersi aldığını, üç kişi ise bir programlama dersi aldığını belirtmiştir.

Veri Toplama

Katılımcıların her biri için en az birer saatlik zaman ayrılarak veri toplama süreci planlanmıştır. Daha sonra her bir katılımcı ile Python derleyicisi ve ekran kayıt programı olan bir bilgisayarın bulunduğu sessiz bir odada görüşülmüştür. Öncelikle katılımcılara araştırmanın amacı ve kendilerinden beklenilenler, sesli düşünme etkinliğinde dikkat edilmesi gerekenler anlatılmıştır. Katılımcılardan beklenilenler kendilerine şu şekilde ifade edilmiştir:

"*Sizden beklenilen sorunun çözümü sırasında aklınızdan geçenleri sesli olarak ifade etmenizdir. Herhangi bir ifadenizin doğru, yanlış, anlamlı ya da anlamsız olması söz konusu değildir. Burada esas olan düşüncelerinizi dile getirmenizdir. Bu düşünceleri savunmanız, nedenlerini açıklamanız ya da doğruluğu ya da yanlışlığı hakkında kaygılanmanız gerekmiyor. Program okuma sorularında elinizde ki kağıdı program yazma sorusunda da bilgisayarı kullanabilirsiniz. İhtiyaç duyduğunuzda internette araştırma yapabilirsiniz.*"

Araştırmada kullanılan sorularla veri toplamaya başlamadan önce katılımcılara örnek bir soru verilerek sesli düşünme ile problem çözme denemesi yapılmıştır. Daha sonra katılımcıya sorular verilerek ses kayıt cihazı ile kod okuma soruları sürecindeki düşünceler, ekran ve ses kayıt programı ile de kod yazma sorusunu çözmeye çalışan katılımcının düşünceleri kayıt edilmiştir. 30-40 saniyeden uzun süren sessizliklerde

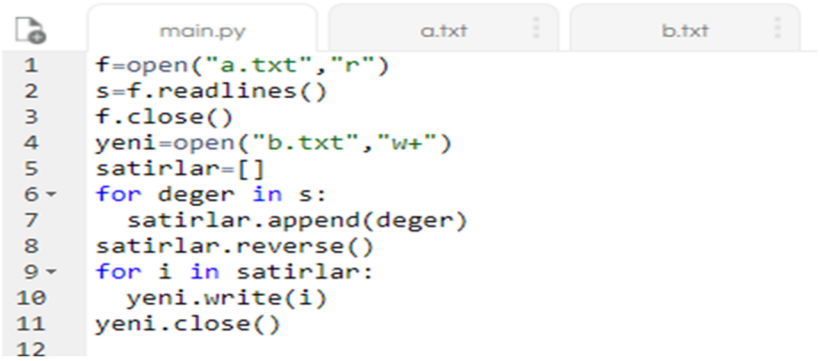
katılımcılar "Sesli düşünmeye çalışır mısın?", "İlgili ilgisiz aklından geçen her şeyi söyleyebilirsin" gibi cümlelerle sesli düşünmeye özendirilmiştir.

Görevler. Bu çalışmada öğrencilere iki tane kod okuma görevi verilmiştir. Bunlardan ilkinde öğrencinin her satırdaki işlemi takip etmesi (Bu soru kod okuma sorusu olarak adlandırılmıştır), ikincisinde ise kodun (programın) işlevini açıklaması istenmiştir (Bu soru programın işlevini açıklama sorusu olarak adlandırılmıştır.). Kod okuma görevi olarak daha önce Teague, Corney, Ahadi ve Lister (2013) tarafından Avusturalya'da IT alanında lisans eğitimi alan ve en az 2 dönem programlama dersi alan öğrencilerin yeni Piagetçi kurama göre mantık yürütme düzeylerini belirlemek için verilen kod kullanılmıştır (Şekil 1). Yeni Piagetçi kurama göre yazılı kodu satır satır takip edememe ya da güçlükle takip edebilme öğrencinin duyusal motor düzeyde olduğunun bir göstergesi olarak kabul edilmektedir (Teague, Corney, Ahadi ve Lister, 2013).

```
1  a=7
2  b=3
3  c=2
4  d=4
5  e=a
6  a=b
7  b=c
8  c=d
9  print(e,a,b,c,d)
```

Şekil 1. Kod okuma görevi

Öğrencilerin kod hakkında mantık yürütme ve kodun bir bütün olarak işlevini sözel olarak ifade edebilme becerilerini ölçmek için Şekil 2'deki kod kullanılmıştır. Yeni Piagetçi kurama göre yazılı kodun işlevini anlama ve sözel olarak ifade edebilme konusundaki yetersizlikler öğrencinin en fazla işlem öncesi dönemde olabileceğinin kanıtı olarak görülmektedir. Bu dönemdeki öğrenci kod okurken fark ettiği girdi ve çıktı değerlerini kullanarak tümevarım yöntemiyle programın işlevi hakkında fikir yürütebilir. Ancak programın bütününe bakıp gerçekleşen hesaplama (computation) ile ilgili tümden gelimci çıkarımda bulunamazlar. Çünkü bu dönemdeki öğrenci programın farklı satırları arasında ilişki kurabilecek düzeyde değildir (Teague, Corney, Ahadi ve Lister, 2013). Dolayısıyla bu çalışmada programlama öğrencilerinin düzeylerini belirlemek için sorulara verdikleri yanıtların yanı sıra bu süreçte sesli düşünme yöntemiyle kayıt altına alınan düşünceleri de göz önüne alınmıştır.



Şekil 2. Program işlevini anlatma görevi

Somut işlemler döneminden soyut işlemler dönemine geçmekte olan ya da soyut işlemler döneminde olan öğrencilerin belirlenmesi için öğrencilerin problemi alt görevlere ayırıp çözüm oluşturmalarını ve çözümlerini bütünleştirmelerini gerektiren bir program yazma görevi oluşturulmuştur (Şekil 3). Bu görev aynı zamanda öğrencilerin programlama davranışlarının incelenmesi için kullanılmıştır. Soyut işlemler dönemindeki programcı, kod yazarken deneme yanılma gibi düşük düzey stratejiler yerine eylem içinde yansıtıcı düşünür. Yazdığı kodun davranışı hakkında öngörüde bulunur ve daha sonra bu öngörüsünü sınar. Hipotetik tümdengelim olarak bilinen bu yöntem soyut işlemler döneminin temel özelliğidir. Bu düzeydeki öğrenci, problemi uygun alt-problemlere parçalama ve görev yapısındaki karmaşa ile baş etme konusunda yetkinleşmiştir (Lister, 2011). Program yazma görevi öğrencilere şöyle ifade edilmiştir: "Kalabalık bir sınıfın iki şubeye bölünmesi planlanmaktadır. Bölme işlemi sonunda tek numaralı öğrenciler a şubesinde, çift numaralı öğrenciler b şubesinde toplanacaktır. Öğrenci numaraları 117001 ile 117099 arasındadır. Sorumlunun klavyeden girdiği "No", "Ad" ve "Soyad" bilgilerini kullanarak a şubesinin "No", "Ad" ve "Soyad" dan oluşan numara sırasına göre oluşturulmuş listesini a.txt belgesi b şubesinin listesini b.txt belgesinde oluşturunuz."

Verilerin Analizi

İlk olarak öğrencilerin kod okuma ve yazma sorularına verdikleri yanıtlar incelenmiştir. Daha sonra ses kayıtları yazılı hale getirilmiştir. Katılımcıların yeni Piagetçi kuram (Lister, 2011; Teague, 2015) çerçevesinde soyutlama düzeylerinin belirlenebilmesi için kuramı oluşturan Raymond Lister'in yeni Piagetçi kurama göre soyutlama beceri düzeylerine ilişkin tanımlamaları temel alınmıştır. Bu temelden yola çıkarak yapılan diğer çalışmalar da incelenerek (Corney, Teague, Ahadi ve Lister, 2012; Falkner, Vivian ve Falkner, 2013; Kutscha, 2017; Lister, 2011, Teague, 2015; Teague, Corney, Ahadi ve Lister, 2013; Teague, Corney, Fidge, Roggenkamp, Ahadi ve Lister, 2012; Teague ve Lister, 2014a, 2014b) soyutlama düzeylerine ilişkin kodlama şeması oluşturulmuştur (Tablo 1). Daha sonra programlama öğretimi

alanında deneyimli bir başka araştırmacı ve yazar, dörder öğrencinin protokollerini yukarıda belirtilen alanyazından yola çıkılarak oluşturulan (Tablo 1.) kodlama şablonuna göre ayrı ayrı değerlendirmişler, çok az sayıda olan farklı değerlendirmeleri tartışarak ortak bir fikre ulaşmışlardır. Diğer protokoller yazar tarafından bu bakış açısıyla kodlanmıştır:

Tablo 1.
Yeni Piagetçi Soyutlama Düzeyleri için Kodlama Şablonu

Duyusal motor dönem	**İşlem öncesi dönem**
• Kod takip etmede zorlanma (A1) • Kavramlarla ilgili yetersiz bilgi (A2) • Kod takip edememe(A3)	• Tümevarımcı (B1) • Girdi çıktı üzerinden programın işlevini belirleme (B2) • Bütüncül bakış açısına sahip olmama (B3) • Adım adım kod okuma (B 4) • Kod ile diagram arasında ilişki kuramama (B5) • Satırları ayrı ayrı düşünme (B6) • Programın işlevini belirlemede zorlanma (B7)
Somut düşünme dönemi	**Soyut düşünme dönem**
• Programdaki soyutlamalarla ilgili fikir yürütme (gerçek durumlarla sınırlı) (C1) • Kod ile ilgili diagram oluşturma (C2) • Geri dönüşlü süreçler hakkında fikir yürütme (C3) • Korunan değerleri belirleme (C4) • Satırlar arası basit ilişkiler kurma (C5) • Tanıdık örüntüleri bulma (C6)	• Yansıtma (D1) • Bütüncül bakış açısı oluşturma (D2) • Karmaşık soyutlamalar oluşturma ve bunları çözüme dönüştürme (D3) • Hipotetik tümden gelimci fikir yürütme (D4)

Katılımcıların program yazma görevindeki programlama davranışları ise bu çalışmanın yazarı ve programlama öğretimi konusunda deneyimli bir başka araştırmacı tarafından sesli düşünme verilerinin ekran kayıtlarıyla birlikte incelenmesi sonucunda oluşturulmuştur. Protokoller katılımcının eylemleri üzerinden parçalara ayrılmıştır (İnternette arama yapma, hatayla karşılaşma, plan yapma, geri dönme, duraksama vb.). Kategori ve kodlar yazar tarafından tüm veri kaynaklarının incelenmesi sonucunda belirlendikten sonra, diğer araştırmacı tarafından da dört öğrencinin program yazma sürecine ilişkin sesli düşünme ve derleyici kayıtları incelenmiş bu süreç sonucunda kod ve kategorilere son hali verilmiştir (Tablo 2).

Tablo 2.
Programlama Davranışlarına İlişkin Kodlama Tablosu

Kategori	**Kod**	**Duyusal motor**	**İşlem öncesi**	**Somut işlemler**
Planlama	Alt görevler oluşturma		2	4
	İşlem temelinde düşünme	1	4	1
	Plan yapmama	1		
Bilgi arama	İşlem temelinde arama yapma		4(10)	2(4)
	Yöntem, hazır fonksiyon vb. arama		2(3)	3(4)
	Yazım kuralı arama		3(4)	2(5)
	Hata mesajı arama			3
Batma anları	Öğrencinin kod yazma sürecinde çıkmaza girdiğini düşündüğü anlar	2	6(10)	5(7)
Karar anları	Kod yazmayı bırakma (pes etme)	2	6	3
	Devam etme		1(2)	
	Kurcalama		1(2)	1(2)
	Yansıtıcı düşünerek yeniden planlama			1(2)
Hata mesajları	Hata mesajı okumama		3	1(3)
	Hatanın programın mantığı ile ilgili olduğunu düşünme		1(2)	
	Yalnızca hatalı satır numarasını okuma		1(2)	2(7)
	Hata mesajını tam olarak okuma			2(5)
Test etme	Alt görevi test etme		2(2)	1(3)
	Alt görevi hipotetik durumlarla test etme			1(3)
	Program test etme			2(2)
	Programı hipotetik durumlarla test etme			

Tabloda sayılar bir kodun kaç katılımcının sesli düşünme kayıtlarında yer aldığını, parantez içindeki sayılar da kaç kez yer aldığını göstermektedir. Örneğin işlem öncesi dönemdeki 4 öğrenci toplam 10 kez işlem temelinde arama yapmıştır. Bu durum 4(10) biçiminde ifade edilmiştir. Daha sonra tüm katılımcılarının sesli düşünme kayıtları yazar tarafından eldeki kodlama şemasına göre kodlanmıştır.

Bulgular

Araştırmaya katılan 13 katılımcının öncelikle sorulara verdikleri yanıtlar incelenmiştir. İki öğrencinin hiçbir soru için kabul edilebilir yanıt oluşturamadıkları, 11 öğrencinin ise kod okuma sorusunu doğru olarak yanıtladıkları belirlenmiştir. Programın işlevini açıklama sorusunu altı öğrenci, program yazma sorusunu ise iki öğrenci doğru olarak yanıtlamıştır (Tablo 3).

Tablo 3
Öğrencilerin Görevleri Tamamlama Durumu

Öğrenci Adı	Görev 1	Görev 2	Görev 3
Öğr_1			
Öğr_2	x	x	x
Öğr_3	x		
Öğr_4	x		
Öğr_5	x	x	
Öğr_6	x	x	
Öğr_7	x		
Öğr_8	x		
Öğr_9			
Öğr_10	x	x	
Öğr_11	x	x	
Öğr_12	x		
Öğr_13	x	x	x

Kod okuma ve yazma sorularına ilişkin yanıtların tek başına öğrencilerin soyutlama becerisi düzeylerini belirlemek için kullanılması yanıltıcı olabilir. Çünkü öğrencinin herhangi bir soruyu nasıl çözdüğü ile ilgili birçok varsayım geliştirilebilir (Teague, Corney, Ahadi ve Lister, 2013). Öğrencilerin problem çözme sürecinde sesli düşünme kayıtlarının protokol analizi yöntemiyle incelenmesi sonucunda sınav sonuçlarından yola çıkılarak elde edilebilecek sınıflamaya oldukça yakın bir sınıflandırma elde edilmiştir. Bu kısımda öncelikle kod okuma ve programın işlevini açıklama sorularına ilişkin analizlere göre soyutlama düzeylerine ilişkin bulgular, daha sonrada belirlenen soyutlama düzeyindeki öğrencilerin programlama davranışlarına ilişkin bulgular paylaşılmıştır.

Duyusal Motor Dönem

Kod okuma sorusunu yanıtlayamayan Öğr_1 ve Öğr_9'un protokol analizi sonucunda da yeni Piagetçi kuramın duyusal motor dönem olarak adlandırdığı gelişim döneminde oldukları görülmüştür. Öğr_1'in atama operatörü konusunda bilgi eksiliği olduğu, atama operatörünü matematiksel bir operatör olan eşittir ile karıştırdığı görülmüştür. Öğr_1 sağ taraftaki değeri, sol taraftaki değişkene atamak yerine operatörün her iki tarafında değişken ismi yer aldığında soldaki değişkenin değerini sağdaki değişkene geçirmiştir. Öğr_1 daha sonra programın işlevi sorusuna geçmiş ve ilk üç satırı doğru bir biçimde anlamlandırmıştır. Ancak dördüncü satırdan sonra kafasının karıştığını, soruyu yanıtlayamayacağını belirtmiştir. İlgili alanyazından yararlanılarak oluşturulan kodlama şemasında duyusal motor dönemin özellikleri

arasında yer alan kavramlarla ilgili yeterli bilgiye sahip olmama, kod takip etmede zorlanma gibi davranışların Öğr_1 ve Öğr_9'un özellikleriyle örtüştüğü görülmektedir.

Öğr_9 ise kod okuma sorusunu ilk birkaç satırda kodu satır satır takip etmeye çalışmıştır. Ancak satır sayısı arttığında kafası karışmış geri dönmüştür. Birkaç kez tekrarlanan sürecin sonunda kod okuma sorusunu yapamayacağını belirtmiştir.

Programın işlevini açıklama sorusunda Öğr_9'un Python programlama dilinin yöntemleri ile ilgili bilgi eksikliğinin olduğu, bu durumun programı anlamasını olanaksız hale getirdiği görülmüştür. Öğr_9'un programın işleviyle ilgili soruya ilişkin sesli düşünme kayıtları, bilgi eksikliğinin sorunun çözümünü nasıl karmaşık hale getirdiğini açık bir biçimde ortaya koymaktadır:

"*Burada başta…a dosyasını açmış, sonra tek tek şey yapmış…readlines ile alta indirmiş sanırım, sonra kapatmış. Sonra b dosyasını yazdırıyor sanırım… şey açmış neydi bu..(uzun bir duraksama) bu append şeysi değeri atama yok satırların içerisindekini…atamaya yarıyordu."*

Her iki öğrenci de başlangıçta program yazma görevini yapmayı deneyebileceklerini belirtmişlerdir. Herhangi bir kod yazmadan birkaç dakika düşünmüşler daha sonra dosya açma ve klavyeden değer almak için gerekli kodları yazmaya çalışmışlardır. Daha sonra her iki öğrenci de benzer biçimde araştırmadan ya da destek olmadan diğer kısımları yazamayacaklarını belirtmişlerdir.

Tablo 4.

Yeni Piagetçi Kurama Göre Kod Okuma Sorularına İlişkin Analiz Sonuçları

Öğrenci Adı	**Kod**	**Dönem**
Öğr_1	A3 A2	Duyusal Motor
Öğr_2	D1 D2 C5 C6	Somut İşlemler
Öğr_3	B3 B4 B6 B7	İşlem Öncesi
Öğr_4	B4 B6 b7	İşlem Öncesi
Öğr_5	C3 D1 C6 C5	Somut İşlemler
Öğr_6	B1 B2 B3 B4 B6 C6	İşlem Öncesi
Öğr_7	A1 B1 B4 B6 B7	İşlem Öncesi
Öğr_8	B1 B3 B4 B6 b7	İşlem Öncesi
Öğr_9	A1 A2	Duyusal Motor
Öğr_10	C3 C5 C6	Somut İşlemler
Öğr_11	C3 C5 C6 D1	Somut İşlemler
Öğr_12	B1 B3 B6 B7 B4	İşlem Öncesi
Öğr_13	C6 D1 C5	Somut İşlemler

İşlem Öncesi Dönem

Kod okuma sorusunu doğru yanıtlayıp programın işlevi sorusunu yanıtlayamayan beş öğrencinin dışında protokol analizi sonucunda her iki soruya da yanıt veren Öğr_6'nın düşünme biçiminin de işlem öncesi dönemin özellikleriyle uyumlu olduğu görülmüştür (Tablo 2).

Bu dönemde yer alan altı öğrencinin tamamı, program açıklama sorusunu satır satır okuyarak programın işlevini anlamaya çalışmışlardır. Yine bu öğrencilerin tamamı, satırların tek başına bağımsız bir iş yaptığı ilk beş satırı zorlanmadan anlamışlardır. Ancak ilk döngünün başlamasıyla beraber altı katılımcıdan dördü programın karmaşıklaştığını belirtip duraksamışlardır. Öğr_4 bunu şu şekilde dile getirmiştir:

"...buraya kadar işler kolaydı…s'nin içindeki değerler için….s neydi….hımm evet a belgesindeki satırlar. Bunları boş listeye atıyoruz…yani for döngüsü içinde satirlara değer atıyoruz bunlarda listeye gidiyor. Sonra reverse, reverse ne yapıyordu….evet çeviriyor bu listeyi….ekrana yazdırıyor sonra yeniden for döngüsü…sanırım bu program bir şekilde listeye bir şeyler yazdırıyor."

Öğr_4'ün işlem öncesi dönemin özelliklerine uygun bir biçimde kodu adım adım takip edebildiği, bu yolla kodun genel işlevi hakkında fikir yürüttüğü ancak kodun içerisinde mantıksal işlemler arttıkça düşüncelerinin daha karmaşık hale geldiği görülmektedir. Benzer bir biçimde Öğr_3 programın işlevi sorusunun ilk 5 satırını adım adım ve hatasız bir biçimde takip etmiştir. Ancak for döngüsüyle beraber daha çok duraksamaya başlamıştır. İlk for döngüsünü üç kez adım adım tekrar etme gereği duymuştur. Liste isminin satirlar olması Öğr_3 için ciddi bir karmaşa kaynağı olmuştur. "satirlar.reverse ()" komutunun gerçek işlevi metin belgesinin her bir satırını bir eleman olarak tutan satirlar listesini ters döndürmektir. Yani son eleman ilk eleman, sondan ikinci eleman ikinci eleman olacak biçimde yeniden sıralamaktır. Ancak Öğr_3 daha önce metin belgesindeki her bir satırın bir eleman olarak listeye atandığını ifade ettiği halde burada her bir satırın kelime kelime bölündüğünü ve o şekilde satırın tersten yazdırıldığını düşünmeye başlamıştır.

"*Burda dosyayı açmış okuyor, okuduktan sonra bu dosyanın yazılması için satır halinde yazılması için s'ye atıyor. Sonra dosyayı kapatmış. Daha sonra bir tane daha dosya açıyor, yazma yapmak için. Liste tanımlamış burada satırlar diye sonra değer diye ekleme yapıyor satırlara değeri ekliyor. S'nin içerisinde değer varsa. Daha sonra satırlar …bu değeri ters çeviriyo, satırlar… satırları daha sonra satırları yazdırıyor….daha sonra…. bi saniye şimdi for'un içinde dönüyor…listeye ekliyor..evet…tamam tamam bi daha bakayım evet satırlara ekliyor daha sonra satırları varsa i'yi yazdıracak…yani her satırı tersten yazdıracak."*

Programlama Davranışları

Bu düzeydeki öğrencilerin program yazma deneyimlerine ilişkin sesli düşünme verileri protokol analizi ile incelenmiştir. Analiz sonucunda bu düzeydeki öğrencilerin

planlama aşamasına çok kısa (30 sn ile 1.20 sn arasında) zaman ayırdığı görülmektedir. Kod yazmaya başlayan öğrencilerin yalnızca ikisi sınırlı düzeyde de olsa görevi alt görevlere bölerek planlama yapmışlardır. Diğer öğrenciler görevi tek bir yapı olarak algılamışlar, nasıl dosya açacakları, kaç dosya açacakları, tek sayıları nasıl bulduracakları vb. adımlarla ilgili düşünceler üretmemişlerdir. Altı öğrenciden dördü program yazma görevini tamamlayabileceği düşüncesiyle göreve başlamışlardır.

Bu düzeydeki öğrencilerin kod yazma sürecindeki bilgi arama davranışları incelendiğinde altı öğrenciden dördünün toplam 10 kez bir işin nasıl yapılabileceğine ilişkin arama yaptığı görülmüştür. Bu öğrenciler "Python tek sayı bulma", "Python dosya satır sıralama" gibi arama terimleri kullanarak hazır kod parçalarına ulaşmaya çalışmışlardır. Bu düzeydeki iki öğrenci "sort" yönteminin metin belgesinde kullanılabileceğini düşünerek "sort python" anahtar kelimeleriyle arama gerçekleştirmiştir. Üç öğrenci ise tür dönüşümü, dosya açma, aralık belirleme, koşul oluşturma konularında toplam dört arama gerçekleştirmişlerdir.

Blikstein vd. (2014), öğrencilerin program geliştirme süreçlerinde bazen aşmakta zorlandıkları sorunlarla karşılaştıklarını bu durumda aldıkları kararların ya da davranış biçimlerinin öğrenci başarısı üzerinde etkili olduğunu belirtmişlerdir. Öğrencilerin kod yazma sürecinde açmaza düştükleri noktalar incelendiğinde, bir öğrencinin klavyeden girilen numara değeri ile tür dönüşümü uygulamadan aritmetik işlem yapmak istediği ve hata mesajını okumadığı için çıkmaza düştüğü görülmüştür. Bu öğrenci kodun farklı yerlerinde düzenlemeler yaparak iki kez kodu çalıştırmayı denemiş ve daha sonra kod yazmaktan vazgeçmiştir. İki öğrenci tek ve çift numaraları belirlediği halde ilgili koşul cümlesini oluşturma aşamasında sorun yaşamışlardır. Her iki öğrenci de internette bu konuda arama yapıp hazır kod bulmaya çalışmışlar, sonuca ulaşamayınca kod yazma görevini sonlandırmışlardır. Bir öğrenci ise dosyalara ilgili bilgileri doğru biçimde yazdırmış ancak numara kontrolü ve dosya içeriğini numara sırasına koyma işlemini yapamamıştır. Bu düzeydeki öğrencilerden yalnızca ikisi hata mesajlarını değerlendirmişlerdir. Bunlardan biri hata mesajının programın algoritması ile ilgili olabileceğini düşünerek programı kontrol etmiş diğer öğrenci ise hata mesajında belirtilen satıra giderek hatanın nedenini bulmaya çalışmıştır. Öğrencilerden hiçbiri program yazma görevini tamamlayamadığı için programı test eden kimse olmamıştır. Ancak dosyaya yazdırma aşamasına gelebilen iki öğrenci yazdıkları kodun tek ve çift numaralı öğrencileri farklı dosyalara yazıp yazmadığını test etmiştir.

Somut İşlemler Dönemi

Kod okuma ve programın işlevini açıklama sorularına verilen yanıtların ve sesli düşünme verilerinin incelenmesi sonucuna Öğr_2, Öğr_5, Öğr_10, Öğr_11 ve Öğr_13'ün program okuma davranışlarının Lister (2011) tarafından programlama öğrencilerinin soyutlama becerilerini sınıflandırmak amacıyla oluşturulan yeni Piagetçi kuramda tarif edilen somut işlemler dönemine karşı geldiği değerlendirmiştir. Bu dönemde yer alan öğrencilerin tamamının programın satırları arasında basit

ilişkiler kurabildiği (C5) ve daha önce sınıfta çözülen problemlerden de yola çıkarak kodun içerisinde örüntüler yakalayabildikleri (C6) ve geri dönüşlü süreçleri zorlanmadan takip edebildikleri (C3) görülmüştür. Örneğin Öğr_2 ilk 4 satırı, teker teker okuyup her bir satırdaki kodun işlevini açıkladıktan sonra kodun 5. satırından 9. satıra kadar olan kısmını "*burada önce bir liste tanımlanıyor, sonra da metin belgesinin içeriği satır satır listeye alınıp ters çevriliyor*" şeklinde açıklamıştır. Benzer bir biçimde Öğr_13 kodun 5. satırdan 8. satıra kadar olan kısmı "*sonra a.txt deki metin satirlar listesine alınıyor*" diye açıklamıştır. Öğr_2 ise programın işlevini açıklama sorusunda kısa bir süre (10-15 sn) kodu inceledikten sonra "*bu kod sanırım bir belgenin içeriğini başka bir belgeye aktarmak için*" şeklinde kodun işlevini açıklamış ancak reverse() yöntemini fark ederek "*tabi bir de reverse işi var*" diyerek kodu baştan satır satır okuyarak kodun işlevini açıklamıştır.

Bu dönemdeki öğrencilerin tamamı kod okuma sorusunu zorlanmadan takip edebilmişlerdir. Öğr_2 ve Öğr_5 kod okuma görevinde kodu satır satır okuyarak değişkenlerin değerlerini belirledikten sonra bu kodun "bir tür kaydırma" işine yaradığını belirtmişlerdir. Bu örneklerden de görülebileceği gibi bu dönemdeki programlama öğrencilerinde kod ile ilgili bütüncül bakış açısı oluşmaya başlamıştır. Araştırmaya katılan bütün öğrenciler yeni Piagetçi kurama göre soyutlama becerisi bakımından ilk üç aşamadaki özellikleri gösterdikleri için program yazma sorusu öğrencileri soyutlama becerisi bakımından sınıflandırmak için kodlanmamıştır.

Programlama Davranışları

Somut işlemler dönemindeki öğrencilerin dördü kod yazmaya başlamadan önce problemi alt problemlere parçaladıklarını gösteren cümleler kurmuşlardır. Örneğin Öğr_5 program yazma görevini okuduktan sonra geri dönüp tekrar okumuş ardından kısa bir sessizlikten sonra "*Tek-çift ayırımı yapmak kolay, numara aralığına göre kontrol edeceğiz… o tamam…dosyaya yazdıracağız…ama dosyada nasıl sıralayacağız?...Orada takılabilirim.*" şeklinde planlama yapmıştır. Öğr_10 ise program yazma görevini 3 kez okumuştur. Görevin bazı kısımlarını okurken yavaşlamış, tekrar etmiş ve bazı kelimelerin altını çizmiştir. Daha sonra kağıt üzerine alt problemleri not etmiştir. Öğr_11 ise görevi iki kez okumuş, üçüncü okuyuşunda sesli düşünmeye başlamıştır. Görevin sonuna geldiğinde "*Buraya kadar kolay ancak son kısmında zorlanabilirim… sort dosya işlemlerinde kullanılıyor mu hatırlayamadım, ona bir bakmam lazım*" şeklinde düşüncelerini ifade etmiştir. Kısacası bu dönemdeki öğrencilerin problemi alt problemlere bölerek plan yapma eğiliminde oldukları görülmüştür.

Bu düzeydeki öğrencilerin program yazma sürecindeki bilgi arama davranışları incelendiğinde işlem temelinde yapılan arama sayısının, işlem öncesi dönemdeki öğrencilere göre daha az olduğu, iki öğrencinin toplam üç kez bu konuda arama yaptığı görülmüştür. Bu aramaların tamamı girdilerin numara sırasına koyulması işlemiyle ilgilidir. Bunun dışında iki öğrenci adlarını bildikleri yöntem ya da hazır fonksiyonların işlevini kontrol etmek için, üç öğrenci ise karşılaştıkları hata mesajlarını anlamak için internette arama yapmışlardır. Karşılaşılan hata mesajlarının

tam olarak okunup internette araştırılması bu dönemin diğer dönemlerden bir diğer farkı olarak nitelendirilebilir. Bu düzeydeki öğrencilerden yalnızca biri hata mesajlarını hiç değerlendirmemiştir. İki öğrenci hata mesajlarını tam olarak okuyarak anlamaya çalışmış, iki öğrenci ise hata mesajındaki satır numarasını okuyarak o satırdaki hatayı bulmaya çalışmıştır.

Öğrencilerin batma anları ve bu anlarda aldıkları kararlar incelendiğinde, Öğr_10 herhangi bir batma anı yaşamadan programı doğru olarak tamamlamayı başarmıştır. Diğer dört öğrenciden ikisi dosyalara tek ve çift numaralı öğrenci bilgilerini yazdırmış ancak dosya içeriklerini sıralama konusunda sorun yaşamışlardır. Öğr_2 bu sorunla karşılaştığında bir süre kullanabileceği hazır fonksiyon ya da yöntem var mı diye araştırma yapmıştır. Daha sonra "*Aslında boşuna uğraşıyorum ben dosyadaki satırları sıralamakla, soruda öyle yazdığı için bende o noktaya kilitlendim kaldım. Aslında bunlara hiç gerek yok. Dosyaya yazmadan önce listeye girip verileri orada basit bir şekilde sıralatabilirdim.*" şeklinde yansıtmada bulunmuş ve görevi söylediği yöntemle tamamlamıştır. Somut işlemler dönemindeki diğer üç öğrenciden Öğr_11, öğrenci numaralarını tek-çift şeklinde ayırma ve yalnızca bir aralıktaki öğrenci numaralarını doğru kabul etme ölçütlerini yerine getirmek için koşul cümlesi yazmaya çalışmış ancak sürekli hata ile karşılaşmıştır. Hata mesajlarını okumayan öğrenci kendisine göre kodda bazı değişiklikler yapmış ancak sonuç alamayınca program yazma görevini bırakmıştır. Somut işlemler dönemindeki öğrencilerin program yazma görevinde daha fazla ilerledikleri dolayısıyla yazdıkları kodları diğer öğrencilere göre çok daha fazla test ettikleri görülmüştür. Programlama görevini tamamlayan iki öğrenciden Öğr_2 programı belirtilen ölçütler içerisindeki değerlerle test ederken, Öğr_10 isim olarak harf dışında karakterlerin girilmesi, boş bırakılması vb. senaryolar karşısında programın davranışını test etmiştir. Alt görevler bakımından değerlendirildiğinde bütün öğrenciler tek ve çift numaralı öğrencileri farklı belgelere kayıt etme, klavyeden değer alma, belgedeki isimleri sıralama gibi alt görevler için kodlarını test etmişlerdir. Öğr_10 alt görevlerle ilgili kodlarını da farklı senaryolar için sınamıştır.

Tartışma, Sonuç ve Öneriler

Bu çalışmanın temel amacı Bilgisayar ve Öğretim Teknolojileri Öğretimi Bölümü öğrencilerinin bilgisayar programlama bağlamında soyutlama düzeylerinin yeni Piagetçi kurama (Lister, 2011) göre değerlendirilmesidir. Bu çalışmada ayrıca farklı düzeylerdeki öğrencilerin programlama davranışlarının incelenmesi de amaçlanmıştır. Çalışmaya katılan 13 öğrencinin kendilerine yöneltilen sorulara verdikleri yazılı yanıtlar yeni Piagetçi kuram göz önünde bulunularak incelenmiştir (Lister, 2011; Teague, 2015). Soruların çözümü ile ilgili sesli düşünme verileri de protokol analizi yöntemiyle incelenmiştir. Protokol analizi sonucunda sağlıklı bir biçimde veri toplanabilen 13 öğrenciden ikisinin duyusal motor düzeyde, altısının işlem öncesi, beşinin de somut işlemler döneminde olduğu görülmüştür. Öğrencilerin program yazma sürecine ilişkin veriler ekran kaydı ve sesli düşünme yöntemi ile

toplanmıştır. Elde edilen veriler protokol analizi ile incelenmiş ve farklı düzeylerdeki öğrencilerin programlama davranışları belirlenmiştir.

Programlama eğitimi yaklaşımları genellikle temel programlama kavramlarının kolaylıkla öğrenildiğini ve öğrencilerin gördükleri kodlar hakkında mantık yürütebilir hale geldiğini varsaymaktadır. Dolayısıyla duyusal motor ve işlem öncesi dönemin çabuk geçildiği kabullenilmektedir. Ancak araştırma sonuçları programlama öğrencilerinin bu dönemi aşmakta, programlama ile ilgili problemlere bütüncül bir çözüm önerisi oluşturmakta zorlandıklarını göstermektedir (Teague ve Lister, 2014). Yapılan araştırmalarda iki dönemden fazla programlama dersi alan öğrencilerin halen duyusal motor ya da işlem öncesi dönemde olabildikleri görülmüştür (Teague, Corney, Ahadi ve Lister, 2013). Bu araştırmada da alanyazındakine benzer bir biçimde lisans eğitimlerindeki son programlama dersini alan 13 öğrenciden yedisinin programlama bağlamında yeni Piagetçi bilişsel gelişim düzeyi, duyusal motor ya da işlem öncesi dönem olarak belirlenmiştir.

Programlama öğretimi alanındaki pek çok çalışma deneyimsiz ve deneyimli programcıların problem çözme, düşünme, kod okuma ve kod yazma süreçleri arasındaki farklılıkların belirlenmesine odaklanmıştır. Bu çalışmalarda çokça vurgulanan noktalardan biri deneyimsiz programcıların kodları satır satır okuma eğiliminde oldukları, anlamlı örüntüler oluşturamadıklarıdır. Bu çalışmada da programın işlevini açıklama sorusunda birçok öğrencinin başlangıçta satır satır kodu anlamlandırdığı ancak döngü içerisinde listeye atama yapma gibi birden fazla yapıyla ilgilenmeleri gerektiğinde kafalarının karıştığı ve hata yaptıkları görülmüştür. Yeni Piagetçi kuramlar soyutlama becerisinin kısa dönem beleğin kapasitesiyle ilgili olduğunu savunmaktadırlar (Morra vd., 2007). Kısa dönem belleğin kapasitesinin yedi artı eksi iki nesneyle sınırlı olduğu bilinmektedir (Miller, 1956). Dolayısıyla öğrencilerin soyutlama becerilerinin farklılaşması bu sınırlı belleği ne kadar etkili kullandıkları ile ilişkilidir. Öğrencinin uzun dönem belleğinde konu ile ilgili bilgiler varsa bu bilgiler kısa dönem belleğe çekilerek tek bir nesne gibi işlenebilir. Dolayısıyla kısa dönem bellekteki sınırlılık yeni öğrenilen bilgilerde daha çok sınırlılık oluşturmaktadır. Bu yaklaşımla örtüşen bir biçimde kodun satırları arasında ilişki kurup örüntüleri algılayamayan öğrencilerin programın işlevini açıklama sorusunda zorlandıkları görülmüştür.

Bu çalışmada elde edilen bulgular deneyimsiz programcıların program okuma ve yazma deneyimleri ile ilgili alanyazındaki çalışma sonuçlarıyla örtüşmektedir. Deneyimsiz programcıların davranış biçimleriyle ilgili çalışmalarıyla tanınan Du Boulay (1989) atama operatörü ile matematiksel eşittir operatörünün karıştırılmasının yeni programcılar arasındaki temel kavram yanılgılarından biri olarak nitelemektedir. Deneyimsiz programcıların temel sorunlarından bir diğeri sahip oldukları bilgi yapılarının eksik olması ve uygulamaya konulabilecek durumda olmamasıdır (Perkins ve Martin, 1986; Winslow, 1996). Bu çalışmada da hem duyusal motor hem de işlem öncesi dönemdeki öğrencilerin sorularda kullanılan programlama yapıları hakkında çoğunlukla bilgi sahibi oldukları ancak bu bilgilerin eksik ve hatalı olduğu görülmüştür. Bu çalışmada duyusal motor dönemdeki bazı öğrenciler atama

operatörüne ilişkin kavram yanılgılarından dolayı kod okuma sorusunu doğru olarak yanıtlayamamışlardır. Öğrencilerin bilgi ve deneyim eksiklikleri kısa dönem belleklerini etkili bir biçimde kullanmalarına olanak vermemektedir. Bu durum bazı öğrencilerin kodların ilk satırlarını başarıyla takip ederken satır sayısı arttıkça ya da satırlar arasındaki ilişki karmaşıklaştıkça kodu takip edememelerine neden olmaktadır.

Yapılandırmacı yaklaşıma göre bireyler, yeni bilgileri daha önceki deneyim ve öğrenmeleriyle edindikleri bilgiler üzerine yapılandırırlar. Teague, Corney, Ahadi ve Lister'e (2013) göre adım adım kod takip etme etkinlikleri soyutlama becerisi gerektirmemektedir. Ancak kod takip etme becerisi kazanmamış bir öğrenci, zihninde programlamaya ilişkin soyutlamalar oluşturamaz. Bu nedenle bir öğrenci kod yazma görevlerinde zorlanıyorsa öncelikle kod takip etme becerisinin gelişip gelişmediği sorgulanmalı ve gerektiği durumda kod takip etme becerisini geliştirebilecek etkinlikler tasarlanmalıdır.

Yeni Piagetçi kuram, öğrencilerin gelişimini sürekli ve sıralı aşamalarla tanımlar. Bu yaklaşıma göre bir önceki aşamadaki gelişimini tamamlayamayan öğrencinin bir sonraki aşamada başarılı olması beklenmez. Teague ve Lister (2014) bu dayanaktan yola çıkarak programlama derslerinde öğrencilerin soyutlama düzeylerinin belirlenip öğretimin buna göre şekillendirilmesi gerektiğini savunmaktadırlar. Bir programın özetlenmesi ya da işlevinin açıklanması öğrencinin soyut düşünme becerisiyle ilişkilendirilmektedir (Clancy ve Linn, 1999). Lister (2011) adım adım kod takip etme becerisini yeni Piagetçi kurama göre işlem öncesi dönemde kazanılan bir beceri olarak tanımlarken, bir programın işlevini açıklayabilmek ile ilgili soruların ancak somut işlemler dönemindeki öğrenciler tarafından çözülebileceğini öne sürmüştür.

Programlama derslerinde genellikle öğrencilerden örnek kod yazması istenmekte, bu şekilde daha fazla kod yazarak öğrencinin iyi bir programcı olması sağlanmaya çalışılmaktadır. Somut ya da soyut işlemler dönemindeki öğrenciler için bu doğru bir yaklaşım olabilir. Ancak işlem öncesi ya da duyusal motor dönemdeki öğrencilerin kod yazma etkinliklerinde kazanımları oldukça sınırlı olacaktır. Duyusal motor, işlem öncesi hatta bazen somut işlemler döneminde öğrencilerin soyutlama becerilerinin geliştirilebilmesi için farklı öğrenme etkinliklerinin tasarlanması gerekmektedir (Lister, 2011).

Murphy, Fitzgerald, Lister ve McCauley (2012), 107 lisans öğrencisinin dönem sonu sınavında, kodun işlevini açıklama ve kod yazma sorularından aldıkları puanları karşılaştırmış ve kod yazma becerisiyle kodun işlevini açıklama becerisi arasında anlamlı bir ilişki olduğunu bulmuştur. Bu çalışmada hem kod takip etme hem de programın işlevini açıklama sorularını doğru olarak yanıtlayan beş öğrenciden ikisi kod yazma görevini de başarıyla gerçekleştirmişlerdir. Diğer üç öğrenci de kod yazma görevini tamamlayamasalar da işlem öncesi ve duyusal motor dönemdeki öğrencilere göre daha anlamlı ve uzun kodlar yazmışlardır.

Öğrencilerin programlama davranışları incelendiğinde duyusal motor ve soyut işlemler dönemindeki öğrencilerin program yazma görevini yerine getirebilecekleri konusunda özgüven duyarak kod yazmaya başladıkları görülmüştür. Ancak bu öğrencilerin önemli kısmı kod yazma görevini alt görevlere bölmeye ya da yazacağı kodu planlamaya çalışmamışlardır. Sonuç olarak bu öğrencilerden hiçbiri program yazma görevinde başarılı olamamıştır. Deneyimsiz programcıların, programı planlamak ve oluşturdukları planı test etmek için çok az vakit ayırdıkları, hata ile karşılaştıklarında programı yeniden değerlendirmek ya da planlamak yerine küçük yerel düzenlemelerle programın düzeltmeye çalıştıkları bilinmektedir (Linn ve Dalbey, 1989).

Problem çözme, program geliştirme sürecinin en önemli bileşenidir. Yapılan pek çok araştırma programlama öğrenmek için öğrencilerin problem çözme konusunda ilgi ve beceri kazanmasının önemini göstermiştir (Deek, Kimmel ve McHugh, 1998; Fincher, 1999; Kay vd., 2000). Alanyazındaki bulgularla örtüşen bir biçimde bu çalışmada problemi alt problemlere ayırarak her bir alt problem üzerinde düşünüp çözüm sürecine başlamayan öğrencilerin kod yazmaya başladığında problemin başlangıçta düşündüğünden daha karmaşık olduğunu anlayıp kod yazmaktan vazgeçtikleri görülmüştür. Güçlükle karşılaştığında kod yazmaya devam eden öğrenciler ise geri dönüp kodu inceleyip var olan kodun eksikliklerini yeniden değerlendirmekten kaçınmışlardır. Öğrenciler bunun yerine internetten benzer kodlar bulma ya da kod üzerinde bazı değişiklikler yaparak hatayı gidermeye çalışmışlar, sonuç alamadıklarında ise vazgeçmişlerdir (Perkins, Hancock, Hobbs, Martin ve Simmons, 1986). İşlem öncesi dönemdeki öğrenciler hata mesajlarını incelemekten kaçınmışlardır. Bu durum hata mesajlarının İngilizce olması dolayısıyla zaten kısa dönem belleği çok verimli kullanamayan işlem öncesi dönemdeki öğrenci için ek bilişsel yük oluşturmasıyla ilişkilendirilebilir.

Eğitsel açıdan bilgi arama var olan bilgileri tamamlamanın ya da bilişsel beceri kazanma sürecinde yardım almanın sık kullanılan bir yoludur (Aleven, Roll, McLaren ve Koedinger, 2016). Bu çalışmada katılımcılar program yazma görevinin farklı aşamalarında farklı konularda internet aramaları gerçekleştirmişlerdir. Duyusal motor düzeyindeki öğrenciler herhangi bir arama yapmamışlardır. Buna karşın işlem öncesi dönemdeki öğrenciler daha çok temel görev ya da alt görevleri gerçekleştiren kodlar için araştırma yaparken somut işlemler dönemindeki öğrencilerin hata mesajlarını anlamak ya da belirli bir işlevi gerçekleştirmek, yazım hatalarını gidermek ya da bazı fonksiyon ve yöntemler hakkında bilgi edinmek için arama yapmışlardır. Eğitsel ortamlarda yardım arama davranışlarına ilişkin çalışmalar doğru zamanda ,doğru düzeyde bilgi ya da yardım arama davranışının öğrenmeyi desteklediğini göstermektedir (Lu ve Hsiao, 2017). Buna karşın bir konuyu öğrenmeye yeni başlayan öğrencilerin fazla deneme yapmadan yardım aramaya başlamasının öğrenmeye katkıda bulunmadığı araştırma bulgularıyla ortaya konulmuştur (Roll, Baker ve Koedinger, 2014). Bu bağlamda somut işlemler dönemindeki öğrencilerin daha az yardıma ihtiyaç duydukları, interneti öğrenmelerini tamamlayıcı bir kaynak olarak kullandıkları görülmektedir. Öte yandan işlem öncesi dönemdeki öğrenciler

internetten örnek kod bulmaya çalışmışlardır. Bu yaklaşımın Lu ve Hsiao'nun (2017) bulgularıyla örtüştüğü görülmektedir.

Bu araştırma, sonuçların genellenmesine olanak sağlayacak bir yöntem ve katılımcı sayısı ile yürütülmemiştir. Ancak katılımcıların eğitimleri sürecinde aldıkları son programlama derslerinde gerçekleştirilen bu çalışmada katılımcılarının hemen hemen yarısının programlama konusundaki bilişsel gelişim düzeyleri işlem öncesi döneme karşılık gelmektedir. Lisans eğitimlerinde öğrencilerin kendilerinden beklenen programlama becerisi düzeyine çıkamadıkları, programlamayı öğrenmekte güçlük çektikleri programlama öğretimine ilişkin alanyazında sıklıkla dile getirilen bir konudur (Clear, Edwards, Lister, Simon, Thompson ve Whalley, 2008). Ancak alanyazındaki çalışmalar çoğunlukla mezun olduklarında IT alanında çalışması beklenen öğrenciler ile gerçekleştirilmiştir. Bu öğrencilerin meslek hayatlarında programlama deneyimi kazanması ve soyutlama becerisi bakımından daha üst düzeylere çıkması beklenebilir. Ancak bu araştırmanın katılımcılarının bilişim teknolojileri öğretmeni olmaları ve öğrencilere hesaplamalı düşünmeyi öğretmenleri beklenmektedir. Dolayısıyla bilişim teknolojileri öğretmeni adaylarının hesaplamalı düşünme becerilerinin geliştirilmesi için bilişim teknolojileri öğretmeni yetiştiren lisans programlarında ilgili derslere daha fazla yer verilmesi gerekmektedir.

Benzer araştırmaların uygun yöntem ve katılımcı sayısı ile yapılması öğretmen adaylarının soyut düşünme becerilerinin, programlama davranışlarının ve bu iki değişken arasındaki ilişkinin anlaşılması açısından alanyazına katkı sağlayabilir. Bunun dışında, öğretmen adaylarının bilgisayar programlama bağlamında soyut düşünme becerilerinin geliştirilmesine yönelik öğretim tasarımlarına ilişkin deneysel çalışmalar ilgili alanyazına katkı sağlayabilir.

Kaynakça

Aleven, V., Roll, I., McLaren, B. M., and Koedinger, K. R. (2016). Help helps, but only so much: Research on help seeking with intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 26*(1), 205–223.

Bennedsen, J., and Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *ACM SIGCSE Bulletin*, *38*(2), 39-43.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., and Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences, 23*(4), 561-599.

Clancy, M. J. and Linn, M. C. (1999). *Patterns and pedagogy*. *SIGCSE '99*. New Orleans, LA, USA, ACM.

Clear, T., Edwards, J., Lister, R., Simon, B., Thompson, E., and Whalley, J. (2008, January). The teaching of novice computer programmers: bringing the scholarly-research approach to Australia. In *Proceedings of the tenth conference on Australasian computing education-Volume 78* (pp. 63-68). Australian Computer Society, Inc.

Corney, M., Teague, D., Ahadi, A., and Lister, R. (2012, January). Some empirical results for neo-Piagetian reasoning in novice programmers and the relationship to code explanation questions. In *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123* (pp. 77-86). Australian Computer Society, Inc.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

de Raadt, M., Toleman, M., and Watson, R. (2004). Training strategic problem solvers. *ACM SIGCSE Bulletin, 36*(2), 48-51.

Deek, F.P., Kimmel, H., and McHugh, J.A. (1998). Pedagogical changes in the delivery of the first-course in computer science: Problem solving, then programming. *Journal of Engineering Education, 87*, 313–320.

Demetriou, A. (Ed.). (1988). *The neo-piagetian theories of cognitive development*: *Toward an integration*. Amsterdam: North Holland.

Ericsson, K. A. and Simon, H. A. 1993, *Protocol analysis*: Verbal Reports as Data Rev. edn (Cambridge, Massachu- setts: MIT Press).

Fincher, S. (1999). What are we doing when we teach programming? 29th ASEE/IEEE Frontiers in Education Conference, 12a4-1–12a4-5.

Gelman, R. (1978). Cognitive development. Annual Review of Psychology, 29, 297-332.

Gluga, R., Kay, J., Lister, R., and Teague, D. (2012, September). On the reliability of classifying programming tasks using a neo-piagetian theory of cognitive development. In *Proceedings of the ninth annual international conference on International computing education research* (pp. 31-38). ACM.

Hanks, B. (2008). Problems encountered by novice pair programmers. *Journal on Educational Resources in Computing (JERIC), 7*(4), 2.

Hughes, J., and Parkes, S. (2003). Trends in the use of verbal protocol analysis in software engineering research. *Behaviour & Information Technology, 22*(2), 127-140.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J., and Crawford, K. (2000). Problem-based learning for foundation computer science courses. Computer Science Education, 10, 109–128.

Linn, M.C., and Dalbey, J. (1989). Cognitive consequences of programming instruction. In E. Soloway & J.C. Spohrer (Eds.), Studying the novice programmer (pp. 57–81). Hillsdale, NJ: Lawrence Erlbaum.

Lister, R. (2011, January). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114* (pp. 9-18). Australian Computer Society, Inc.

Lister, R. F. (2007). The neglected middle novice programmer: Reading and writing without abstracting. *National Advisory Committee on Computing Qualifications*.

Lu, Y., and Hsiao, I. H. (2017). Personalized Information Seeking Assistant (PISA): from programming information seeking to learning. *Information Retrieval Journal*, 1-23.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., ... and Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4), 125-180.

Morra, S., Gobbo, C., Marini, Z., and Sheese, R. (2012). *Cognitive development: neo-Piagetian perspectives*. Psychology Press.

Mow, I. C. (2008). Issues and difficulties in teaching novice computer programming. *Innovative techniques in instruction technology, e-learning, e-assessment, and education*, 199-204.

Murphy, L., Fitzgerald, S., Lister, R., and McCauley, R. (2012, September). Ability to'explain in plain english'linked to proficiency in computer-based programming. In *Proceedings of the ninth annual international conference on International computing education research* (pp. 111-118). ACM.

Özdinç, F., ve Altun, A. (2014). Bilişim teknolojileri öğretmeni adaylarının programlama sürecini etkileyen faktörler. *İlköğretim Online*, 13(4).

Özer Şanal, S. ve Erdem, M. (2017, Mayıs). Kodlama ve Robotik Çalışmalarını Problem Çözme Süreçlerine Etkisi: Sesli Düşünme Protokol Analizi.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., and Simmons, R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research, 2*(1), 37-55.

Perkins, D. N. and Martin, F. (1986). Fragile Knowledge and Neglected Strategies in Novice Programmers. Chapter 15 of Empirical Studies of Programmers. E. Soloway and S. Iyengar. Norwoo, New Jersey, Ablex Publishing Corporation.

Robins, A., Haden, P., and Garner, S. (2006, January). Problem distributions in a CS1 course. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 165-173). Australian Computer Society, Inc..

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.

Roll, I., Baker, R. S. D., Aleven, V., and Koedinger, K. R. (2014). On the benefits of seeking (and avoiding) help in online problem-solving environments. *Journal of the Learning Sciences, 23*(4), 537-560.

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850-858.

Teague, D. (2015). *Neo-Piagetian Theory and the novice programmer* (Doctoral dissertation, Queensland University of Technology).

Teague, D. M., Corney, M. W., Fidge, C. J., Roggenkamp, M. G., Ahadi, A., and Lister, R. (2012, December). Using neo-Piagetian theory, formative in-Class tests and think alouds to better understand student thinking: a preliminary report on computer programming. In *Proceedings of 2012 Australasian Association for Engineering Education (AAEE) Annual Conference*.

Teague, D., and Lister, R. (2014, January). Manifestations of preoperational reasoning on similar programming tasks. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148* (pp. 65-74). Australian Computer Society, Inc.

Teague, D., Corney, M., Ahadi, A., and Lister, R. (2013, January). A qualitative think aloud study of the early neo-piagetian stages of reasoning in novice programmers. In *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136* (pp. 87-95). Australian Computer Society, Inc.

Teague, D., Corney, M., Ahadi, A., and Lister, R. (2013, January). A qualitative think aloud study of the early neo-piagetian stages of reasoning in novice programmers. In *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136* (pp. 87-95). Australian Computer Society, Inc..

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. ACM SIGCSE Bulletin, 28(3), 17-22.

Ankara University Journal of Faculty of Educational Sciences
Year: 2018, Volume: 51, Issue: 1, 1-26
DOI:10.30964/auebfd.404978, E-ISSN: 2458-8342, P-ISSN: 1301-8342

Determining Abstraction Levels and Programming Behaviors of Pre-Service Teachers of Information Technologies According to the Neo-Piagetean Theory

ARTICLE TYPE	Received Date	Accepted Date	Online First Date
Research Article	11.23.2017	03.13.2018	03.14.2018

Hüseyin Özçınar [1]
Pamukkale University

Abstract

The aim of this research is to determine abstraction skills of the information technology teacher candidates according to the new Piagetian theory. Determining theparticipants' programming behaviors according to the determined abstraction levels is another purpose of this research. The research is designed as a qualitative research. Participants of the research consisted of 15 information technologies teacher candidates who enrolled in the Internet Based Programming course in the Department of Computer Education and Instructional Technology and volunteered to take part in the research. The students were asked to answer three questions, one for coding a program, and two for reading codes. The code reading questions were answered on the paper and program coding question was answered on the computer. Students' thinking aloud protocol recodings were the main data source, recordings of students 'code writing processes and written responses to code reading questions were used as auxiliary data source. Two of the students were found to be at psychomotor stage according to the new Piagetian theory, six of the students were at the pre-opperational stage and five of them were at the concrete operational stage.

Keywords: Neo-Piagetian theory,computer programming, programming behaviors

[1]*Corresponding Author*: Assist. Prof. Dr., Faculty of Education, Department of Computer and Instructional Technologies Education, hozcinar@pau.edu.tr, https://orcid.org/0000-0001-8715-2653

Summary

Purpose and Significance

The acquisition of abstraction skills in a problem area does not occur at the same rate for each student in the classroom. Within the same class, some students can be in the preoperational stage while others are in the stage of concrete operational or formal operational. In this case, expecting each student in the class to write codes in parallel with the expectations of the teacher may reduce the success of the programming courses. It will be a more successful approach to determine the new Piagetian development stages of the students and to ensure that abstraction skills are acquired through appropriate approaches to these stages (Lister, 2011). In this context, this research aims to determine the abstraction skills of pre-service teachers of information technology according to the new Piagetian theory. Another aim of the research is to determine the participants' programming behaviors according to the abstraction levels.

Method

The research was designed as a qualitative research. The participants of the study consisted of 15 pre-service teachers of information technologies who had taken the Internet Based Programming course in the Department of Computer and Instructional Technology Education at the Faculty of Education and voluntarily accepted to participate in the research. The pre-service teachers were asked to answer three questions consisting of two code-reading and one code-writing questions by thinking out loud. The code- reading questions were answered on paper, and code-writing questions were answered on computer. The audio recordings of the students were used as the major data source while the screen recordings of the students' code- writing processes and written answers given to the code-reading questions were used as the auxiliary data sources.

Results

It was observed that two of the students who participated in the research were on sensorymotor stage according to the new Piagetian model, six of them were in the stage of preoperational and five of them were in the concrete operational stage. In this study, students in the sensorymotor stage could not answer the code-reading question correctly due to misconceptions about the assignment operator. It was observed that the students in the preoperational stage allocated very little time for planning. Students at this level found sample codes on the internet and tried to accomplish the task assigned to them. None of the students at this level could successfully complete the task of code-writing. Students in the concrete operational stage successfully answered the questions of code-reading and explaining the program function. A significant number of students at this level attempted to separate the problem into sub-problems before writing code. These students spent more time to make sense of error messages. Two of the five students in the concrete operational stage were able to complete the task of program-writing.

Discussion and Conclusions

It is known that inexperienced programmers have little time to plan and test the program, and they try to fix the program through small local edits instead of re-evaluating or planning the program (Linn and Dalbey, 1989). In parallel with the findings in the literature, this study shows that the students, who started writing code without separating the problem into sub-problems to reflect on each sub-problem in order to start the resolution process, understood that the problem was more complicated than originally thought and they ultimately gave up writing the code. The students who continued to write code when faced with difficulty, on the other hand, avoided to go back to examine the code and re-evaluate the missing points of the code. Instead, they attempted to find similar codes on the internet or to make some changes on the code to get rid of the error (Perkins, Hancock, Hobbs, Martin and Simmons, 1986). Students in the pre operational stage refrained from examining error messages. This can be attributed to the fact that the error messages were in English so that they already had an additional cognitive burden for the student in the preoperational stage, who cannot use the short term memory very efficiently. In this study, the participants made different internet searches in different stages of the program-writing task. The students at the sensorymotor level did not make any search. On the other hand, those in the preoperational stage searched for codes that perform basic tasks or sub-tasks while those in the concrete operational stage searched to understand the error messages or to perform a specific function, to get rid of their mistakes in writing or to learn about some functions and methods.