eefdergi
Cilt 23 Sayı 3 Erzincan Üniversitesi Eğitim Fakültesi Dergisi
e-ISSN 2148-7510
https://dergipark.org.tr/en/pub/erziefd | eefdergi@erzincan.edu.tr
Doi numarası: 10.17556/erziefd. 815922

Test Development for Debugging Performance: Validity and Reliability Study*

Arif AKÇAY*, Arif ALTUN**

Received date: 24.10.2020 **Accepted date: 07.12.2020**

Abstract

Programming is to produce computer-based solutions for a problem by situation analysis, algorithm and program design, implementation, testing and debugging. It is usual situation that encounter bugs in the programming process. The process of resolving the bugs and making the generated program codes free of bugs is called debugging. Debugging requires deeper knowledge than writing new code. Debugging is a skill that programmers must acquire, like good coding and write efficient code. In the literature, there is a lack of a valid and reliable measurement tool for the evaluation of debugging performances. In this study, it is aimed to develop an achievement test in order to evaluate the debugging performance of students studying in the Information Technologies departments of vocational high schools. During development of the test, the purpose of the test, the bugs encountered and their scope were determined, related literature reviewed, the test consisting of 27 code scenarios was created and expert opinion was obtained to ensure content validity. After expert opinions and pre-pilot implementation, 18 code scenarios were determined for pilot implementation. 148 students from different vocational and technical high schools in the Information Technology Department participated in the pilot implementation of the test. According to the analysis carried out for the test and the items, the necessary items were removed. As a result, the item difficulty indexes are between .30 and .49, the item discrimination indexes are between .39 and .69, and the item reliability indexes are between .20 and .33 in the test. Also, the KR-20 reliability index of test is .575. Therefore, it can be said that The Debugging Performance Test is valid and reliable.

Keywords: Programming, debugging performance, compiler-time bugs, run-time bugs, logical bugs.

* This article is based on the Ph.D. thesis written by the first author under the supervision of the second author. This study was presented as an oral presentation at the "4th International Turkish Computer and Mathematics Education Symposium" on September 26-28, 2019.

* Kastamonu University, Computer Education and Instructional Technology Department, Kastamonu, Turkey; aakcay@kastamonu.edu.tr
** Hacettepe University, Computer Education and Instructional Technology Department, Ankara, Turkey; altunar@hacettepe.edu.tr

Hata Ayıklama Performansı Testi Geliştirme: Geçerlik ve Güvenirlik Çalışması*

Arif AKÇAY*, Arif ALTUN**

Geliş tarihi: 24.10.2020 **Kabul tarihi: 07.12.2020**

Öz

Programlama, bir probleme yönelik durum analizi, algoritma ve program tasarımı, uygulama, test etme ve hata ayıklama yaparak bilgisayar tabanlı çözümler üretmektir. Programlama sürecinde içerisinde hata ile karşılaşmak olağan bir durumdur. Hataların çözümlenmesi ve oluşturulan program kodlarının hatalarından arındırarak işlevsel hale getirilmesi işlemi ise hata ayıklama olarak adlandırılmaktadır. Hata ayıklama yeni kodlar yazmaktan daha derin bir bilgi birikimi gerektiren, iyi kodlama yapmak ve verimli kod yazmak gibi programcıların edinmesi gereken bir beceridir. Alanyazında hata ayıklama performanslarının değerlendirilmesine yönelik geçerli ve güvenilir bir ölçme aracının eksikliği görülmüştür. Bu çalışmada, meslek liselerinin Bilişim Teknolojileri bölümlerinde okuyan öğrencilerin hata ayıklama performanslarını değerlendirmek için bir başarı testi geliştirmek amaçlanmıştır. Testin geliştirilmesi aşamasında testin amacı, karşılaşılan hatalar ve kapsamları belirlenmiş, ilgili alanyazın incelenmiş, 27 kod senaryosundan oluşan test yönergesiyle birlikte kapsam geçerliğinin sağlanması amacıyla uzman görüşlerine sunulmuştur. Uzman görüşleri ve ön pilot uygulaması sonucunda 18 kod senaryosu belirlenmiş ve farklı mesleki ve teknik liseleri Bilişim Teknolojileri bölümlerinde öğrenim gören 148 öğrenciyle pilot uygulaması yapılmıştır. Teste ve maddelere yönelik yapılan analizler doğrultusunda gerekli görülen maddeler çıkartılmıştır. Sonuç olarak oluşan test maddelerinin güçlük indeksleri .30 ile .49 arasında, ayırt edicilik indeksleri .39 ile .69 arasında ve madde güvenirlik indeksleri .20 ile .33 arasındadır. KR-20 güvenirlik indeksi .575 olan Hata Ayıklama Performansı Testi (HAPT)'nin geçerli ve güvenilir olduğu söylenebilir.

Anahtar kelimeler: Programlama, hata ayıklama performansı, derleme zamanı hatası, çalışma zamanı hatası, mantık hatası.

* Bu çalışma ikinci yazar danışmanlığında ilk yazar tarafından yapılan doktora tezi kapsamında üretilmiştir. Bu çalışma 26-28 Eylül 2019 tarihlerinde "Uluslararası Türk Bilgisayar ve Matematik Eğitimi Sempozyumu – 4" de sözlü bildiri olarak sunulmuştur.

* Kastamonu Üniversitesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Kastamonu, Türkiye; aakcay@kastamonu.edu.tr

** Hacettepe Üniversitesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Ankara, Türkiye; altunar@hacettepe.edu.tr

1. Giriş

Programlama, bir problemin çözümünü sağlamak için durum analizi, algoritma ve program tasarımı, uygulama, test etme ve hata ayıklama adımlarını içeren bir süreçtir. Bu süreçte karşılaşılan ve süreci olumsuz etkileyen durumlar hata olarak adlandırılmıştır (Downey & Mayfield, 2016). Soloway ve diğerleri yapmış oldukları çalışmalarında programlamayı bir hedef/plan süreci olarak tanımlamış, hedefe ulaşmak için yapılan planların uygulanması sırasında hedefe ulaşmakta engel olan durumları hata olarak ifade etmişlerdir (Soloway & Ehrlich, 1984; Spohrer, Soloway, & Pope, 1985). Programlama süreci hata yapmaya eğimlidir (Downey & Mayfield, 2016). Bu bakımdan programın tasarlanması ve uygulanması sırasında çeşitli hatalara rastlanması yaygın görülen bir durumdur. Programlama sırasında hataların oluşması, hazırlanan programın niteliğinin düşmesine ve programın amacına hizmet etmemesine neden olmaktadır (Peng, Li, Song, Hu, & Feng, 2016). Bu bakımdan hataların bulunması ve çözümlenmesi önemlidir.

Hata bulma ve çözümleme süreçlerin kolaylaştırılması için hataların çeşitli tanımlamaları ve sınıflandırmaları yapılmıştır. Alanyazında her ne kadar hataların bulunduğu kod yapılarına göre (Institute of Electrical and Electronics Engineers, 2010) veya hatalı kodların uygulanma durumlarına göre (Johnson, Soloway, Cutler, & Draper, 1983) sınıflandırmalar bulunsa da, hata türlerine yönelik yapılan tanımlamalar incelendiğinde programlama yaparken karşılaşılan hataların genellikle üçe ayrıldığı görülmektedir (Ahmadzadeh, Elliman, & Higgins, 2005; Downey & Mayfield, 2016; Hristova, Misra, Rutter, & Mercuri, 2003). Bu hata türlerinin ilki yazıldığı dile özgü, dilbilgisi ile ilgili olup, sembol veya karakterlerin eksik olması gibi durumlarda derleme zamanı karşılaşılan hatalardır. Alanyazında bu hatalar her ne kadar sözdizimsel hatalar olarak isimlendirilmiş çalışmalar olsa da bu çalışmada derleme zamanı hataları olarak bahsedilmiştir. Bir diğer hata türü ise çalışma zamanı hatalarıdır. Bu hatalar dilde verilen komutların yanlış yorumlanması ile sonsuz döngüye girmesi gibi çalışma sırasında beklenmedik durumları ifade etmektedir. Son olarak mantık hataları ise kodlama yapılırken programcının yanlış düşüncelerinden kaynaklanan, programın amacına yönelik doğru çıktı vermemesi durumunu ifade etmektedir (Ahmadzadeh ve diğerleri, 2005; Downey & Mayfield, 2016; Hristova ve diğerleri, 2003). Tüm bu hatalar ile programlama süreci içerisinde karşılaşılabilecek olup, oluşturulan programın amacına yönelik çalışması için bu hataların incelenmesi, bulunması ve giderilmesi gereklidir.

Programlama yapılırken hatalar ile karşılaşıldığında bu hataların sebeplerinin incelenmesi, bulunması, düzeltilmesi ve yeniden test edilmesiyle hataların çözümlendiğinden emin olunması adımlarına hata ayıklama denilmektedir (Dooley, 2011). Hata ayıklama, planlanan program ile gerçekleşen program arasındaki farkın anlaşılması, uygulama dilinin bilinmesi, genel programlama ve uygulama alanı hakkında fikir edinilmesi, karşılaşılan hatalar ve hata türleri hakkında bilgi sahibi olunması gerekliliği (Ducasse & Emde, 1988) bakımından yeni kodlar yazmaktan daha derin bir anlayış ve beceri gerektirmektedir (Liu, Zhi, Hicks, & Barnes, 2017). İyi bir hata ayıklayıcı olan birinin programlamada da uzman olacağı, fakat her programlama yapan kişinin iyi bir hata ayıklayıcı olamayacağına ilişkin görüşler (Ahmadzadeh ve diğerleri, 2005; Fitzgerald ve diğerleri, 2008; McCauley ve diğerleri, 2008) bu becerilerin önemine vurgu yapmaktadır. Programlama süreci içerisinde kaliteli ve verimli kod yazmak, hataları fark etmek, düzeltmek ve daha da önemlisi oluşturulmak istenen programın işlevsel hale gelmesi adına bu bilgi ve beceriler gereklidir. Aynı şekilde bu bilgi ve becerilerin profesyonel gelişimlerinin bir

gerekliliği olarak programlama yapan mesleki ve teknik liselerin Bilişim Teknolojileri (BT) öğrencileri için de önemli olduğu söylenebilir.

Mesleki ve teknik liselerin BT bölümlerinde öğrenim gören öğrenciler çağın gerektirdiği yeterliliklerin kazanılması ve ilgili sektörlere iş gücünün sağlanması amacıyla yetiştirilmektedir. İşletmelerin kurumsallaşmaları için BT alanlarında yetişmiş iş gücüne ihtiyaç duyulmakta, bu öğrencilerin gelecekte çalışma hayatının en önemli unsuru olacağı öngörülmektedir (Milli Eğitim Bakanlığı [MEB], 2011b). BT bölümlerinde öğrenim gören öğrencilerin eğitimlerinde programlamanın önemli yeri bulunmaktadır. Bölüm öğrencileri, 10. sınıf düzeyinde haftada 4 saat programlama temelleri dersi görmekte, bu ders kapsamında C# dilinde kodlama uygulamaları gerçekleştirmektedirler. Ayrıca öğrenciler seçtikleri alanlarına göre daha özelleştirilmiş programlama eğitimleri de (ör; nesne tabanlı programlama, web tasarımı ve programlama vb.) almaktadırlar (Milli Eğitim Bakanlığı, 2011b). Bu dersler çerçevesinde MEB, öğrencilere hatalar konusunda bilgi sahibi olmak, hataları kontrol etmek, hata mesajlarını anlamak, hataları fark etmek ve düzeltmek gibi bilgiler ve beceriler kazandırmayı amaçlamaktadır (Milli Eğitim Bakanlığı, 2011f). Alanyazında bu bilgilere ve bu becerilere sahip olan öğrenciler iyi hata ayıklayıcı olarak nitelendirilirken, sahip olmayan öğrenciler ise zayıf hata ayıklayıcılar olarak nitelendirilmektedir (Ahmadzadeh ve diğerleri, 2005). Alanyazında yapılan birden fazla çalışmada bu şekilde nitelendirme yapabilmek, öğrencilerin hata ayıklama performanslarını belirleyebilmek için çeşitli yaklaşımlar bulunmaktadır.

Yapılan çeşitli çalışmalarda öğrencilerin hata ayıklama performansları değerlendirmeye çalışılmıştır. Bu çalışmalarda genel olarak öğrencilerin hata ayıklama performansları değil, hata ayıklarken stratejilerinin ve düşünme süreçlerinin belirlenmesi amaçlanmaktadır (Bednarik, 2012; Grigoreanu ve diğerleri, 2009; Romero, du Boulay, Cox, Lutz, & Bryant, 2007; Yen, Wu, & Lin, 2012). Dolayısıyla hata ayıklama uygulamaları gerçekleştirilirken asıl amacın hata ayıklama performanslarının ölçülmesinin olmadığı görülmektedir. Hata ayıklama uygulamalarıyla değerlendirilen hata ayıklama performansları genellikle çok az sayıda kod senaryosu üzerinden yapılmakta olup, bu kod senaryoları iki ile altı arasında değişmektedir (Bednarik, 2012; Bednarik & Tukiainen, 2004a, 2004b, 2008; Fitzgerald ve diğerleri, 2008; Khan, Brinkman, & Hierons, 2011; Romero, Cox, du Boulay, & Lutz, 2002; Romero, Lutz, Cox, & du Boulay, 2002). Bu bakımdan bir dil yapısında olan kod yapılarını kapsamayacağı veya ilgili hata türlerinin hepsini barındırmayacağı düşünülebilir. Bu durumu destekler nitelikte kod senaryoları içerisinde birden fazla farklı farklı hatalar bulunmaktadır (Bednarik, 2012; Chen, Wu, & Lin, 2013; Fitzgerald ve diğerleri, 2008; Grigoreanu ve diğerleri, 2009; Yen ve diğerleri, 2012). Örneğin, aynı kod senaryosu içerisinde hem dil problemleri, hem yanlış yerleştirilmiş döngü, hem aritmetik hata hem de güncelleme hatası bulunmaktadır (Fitzgerald ve diğerleri, 2008). Tüm bunlar düşünüldüğünde kod senaryolarının içerilerinde barındırdıkları hataların özellikleri ve sayıları farklılaştıkça hata ayıklama performansları üzerinde etkileşim etkisi yaratması muhtemeldir. Yapılan bu çalışmaların katılımcıları genellikle üniversite öğrencileri olup, öğretmenler, öğretim üyeleri, mezun öğrenciler ve bilişim teknolojileri çalışanları da katılım göstermişlerdir (Bednarik, 2012; Chen & Lim, 2013; Grigoreanu ve diğerleri, 2009; Romero ve diğerleri, 2007; Yen ve diğerleri, 2012). Ayrıca araştırmalarda hata ayıklama performanslarını değerlendirmek adına sunulan kod senaryolarında Java dili kullanıldığından dolayı (Bednarik, 2012; Fitzgerald ve diğerleri, 2008; Romero ve diğerleri, 2007) Türkiye'deki mesleki ve teknik liseleri Bilişim Teknolojileri öğrencileri için uygun olmadığı düşünülebilir. Ayrıca tüm bu çalışmaların bazılarında hata ayıklama görevlerinin uzman görüşleri alınarak hazırlandığı belirtilse de (Khan ve diğerleri, 2011) çoğunluğunda bu görevlerin geçerlik ve güvenirlik çalışmalarına yönelik

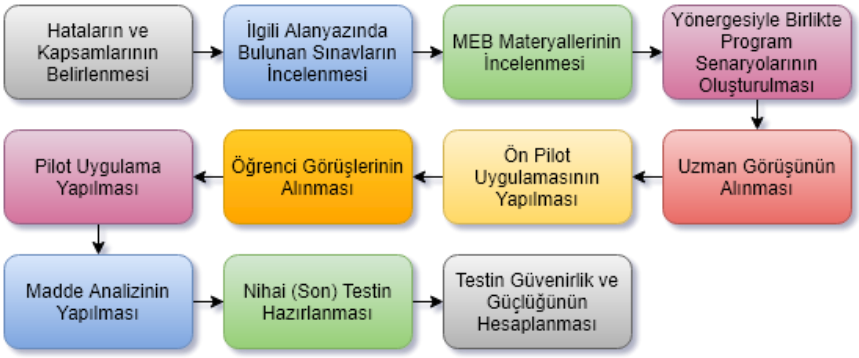
bilgiler sunulmamıştır. Ulusal alanyazın incelendiğinde ise programlamaya ilişkin farklı bilgi ve becerileri ölçmek amacıyla geliştirilen ölçme araçlarında öğrencilerin hata ayıklama performanslarına yönelik öz raporlamaya dayalı, sınırlı madde sayısı olduğu görülmüştür (Altun & Kasalak, 2018; Altun & Mazman, 2012; Yılmaz, 2013). Bu bakımdan alanyazında mesleki ve teknik liselerin BT öğrencilerin hata ayıklama performanslarının değerlendirilmesine yönelik geçerli ve güvenilir bir ölçme aracının eksikliği görülmüştür.

1.1. Araştırmanın Amacı

Bu çalışmanın amacı Türkiye'de MEB bağlı mesleki ve teknik liselerin Bilişim Teknolojileri öğrencilerinin programlama sürecinde karşılaşıyor oldukları hataları ayıklama performanslarının değerlendirilmesini sağlayacak geçerli ve güvenilir başarı testi geliştirmektir. Bu amaç doğrultusunda uygulanan yöntem, elde edilen bulgular, yapılan tartışma ve ulaşılan sonuç başlıklar halinde sunulmuştur.

2. Yöntem

MEB'e bağlı mesleki ve teknik liselerin Bilişim Teknolojileri bölümlerinde öğrenim gören öğrencilerin programlama uygulamalarında karşılaşıyor oldukları hataları ayıklama performanslarının değerlendirilmesi amacıyla hatalar içeren kod senaryoları ve bu hatalara yönelik test soruları oluşturulmak istenmiştir. Bu kod senaryolarının ve ilgili testin oluşturulma süreci Şekil 1'de belirtilmiştir.



Şekil 1. HAPT'in geliştirme sürecinin şeması

Hata Ayıklama Performansı Testi (HAPT) geliştirme sürecinde ilk önce öğrencilerin programlama süreci içerisinde karşılaşabilecekleri hataların ve bu hataların kapsamlarının belirlenmesi amacıyla literatür taraması yapılmıştır. Daha sonra geliştirilmek istenen test ile ilgili alanyazında bulunan uluslararası sınavlar ve görevler ("AP Computer Science A" kursu sınav soruları [Albert, 2018]) incelenmiştir. Oluşturulacak kod senaryolarının öğrencilere uygun olması amacıyla MEB tarafından programlama temelleri dersi için Mesleki Eğitim ve Öğretim Sisteminin Güçlendirilmesi Projesi (MEGEP) kapsamında hazırlanan bireysel öğrenme materyalleri incelenmiştir (Milli Eğitim Bakanlığı, 2011a, 2011c, 2011d, 2011e). Uluslararası farklı sınavlarda her ne kadar farklı dillerde testler bulunsa da (Ör; Java) MEB tarafından hazırlanan materyaller ve yapılan öğretimlerde C# dili kullanıldığı için oluşturulacak kod senaryolarında bu dilin kullanılmasına karar verilmiştir. Uzman görüşüne sunmak amacıyla farklı türde (derleme zamanı hatası, çalışma zamanı hatası ve mantık hatası) ve görece farklı düzeyde (kolay, orta ve zor) hatalar içeren toplam 27 kod senaryosu ve ilgili hatalara yönelik test maddeleri oluşturulmuştur. Farklı türde hatalar içeren 27 kod senaryosu ve ilgili test soruları, testin kapsam geçerliğinin sağlanması amacıyla yönergesiyle birlikte uzman görüşüne sunulmuştur (Alpar, 2014; Baykul, 2010; Karakoç & Dönmez, 2014; Özbek, 2014; Sönmez & Alacapınar, 2016). Uzman görüşleri

sonrasında test düzenlenmiş ve ön pilot uygulaması gerçekleştirilmiştir. Bu uygulama sonrasında ön pilot uygulaması gerçekleştirilen öğrencilerle yapılan görüşmeler doğrultusunda yeniden düzenleme yapılmıştır. Daha sonrasında ise test, pilot uygulama için hazır hale getirilmiş ve uygulanmıştır. Gerekli madde analizleri yapılarak testin son hali belirlenmiş ve oluşan teste ilişkin bilgiler sunulmuştur.

2.1. Katılımcılar

MEB'e bağlı mesleki ve teknik liselerin Bilişim Teknolojileri bölümlerinde görev yapan iki öğretmenden uzman görüşü alınmıştır. Uzmanların birinin 15 yıllık öğretmenlik, 10 yıllık programlama eğitimi verme tecrübesi; diğerinin ise 13 yıllık öğretmenlik ve programlama eğitimi verme tecrübesi ve bir programlama kitabı yazarlığı bulunmaktadır. Uzman görüşleri sonrasında düzenlenen testin ön pilot uygulaması için programlama temelleri dersinde farklı düzeyde başarı gösteren (yüksek, orta ve düşük) üç öğrenci belirlenmiştir. Belirlenen öğrencilerle birlikte ön pilot uygulaması gerçekleştirdikten sonra test hakkında görüşme yapılmıştır. Testin pilot uygulaması için Kastamonu ilinin farklı ilçelerinde bulunan beş farklı mesleki ve teknik lisenin Bilişim Teknolojileri bölümlerinin 10. ve 11. Sınıf düzeyinde okuyan 148 öğrenci (66 erkek, 82 kız) katılım göstermiştir. Çalışma kapsamında belirlenen öğrencilerin 10. ve 11. Sınıf düzeyinde olmalarının nedeni ise bu testi gerçekleştirebilmeleri için gerekli olan programlama temelleri dersinin Bilişim Teknolojileri bölümü 10. sınıf düzeyinde veriliyor olmasıdır (Milli Eğitim Bakanlığı, 2011b).

2.2. Verilerin Toplanması

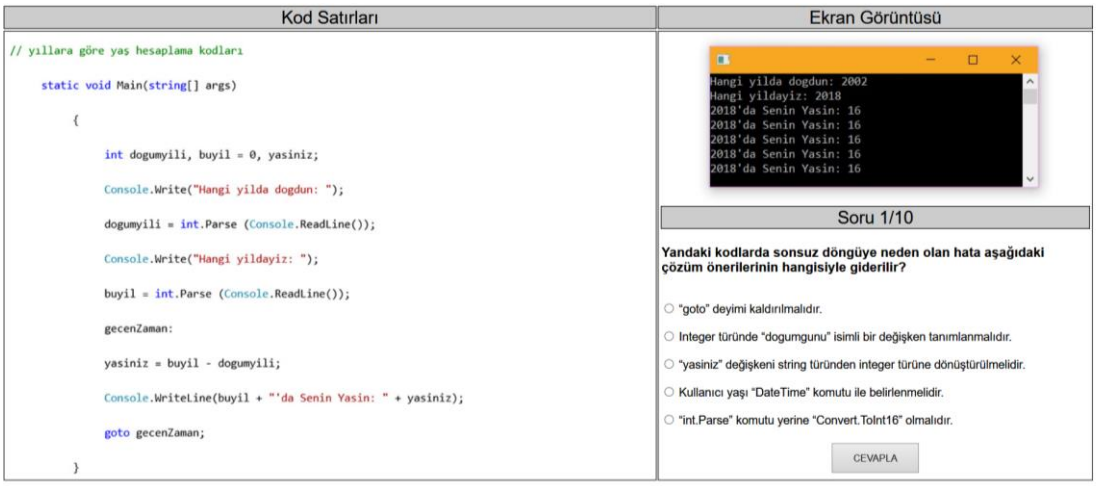
İlk olarak hazırlanan yönerge, kod senaryoları ve test soruları içeren test formu hakkında uzman görüşleri alınmıştır. Uzman görüşleri 5-9 Kasım 2018 tarihleri arasında öğretmenlerin okullarına gidilerek kâğıt-kalem tabanlı form aracılığıyla alınmıştır. Uzman görüşleri alınmadan önce teste ilişkin yönerge sunularak, bilgilendirme yapılmıştır.

Testin ön pilot uygulaması 26 Kasım 2018 tarihinde gerçekleştirilmiştir. Ön pilot uygulaması için ilk önce öğrenciler okullarında bulunan bilgisayar laboratuvarına davet edilmiştir. Ön pilot uygulamasına başlamadan önce öğrencilere bilgilendirme yapılmış ve varsa soruları cevaplandırılmıştır. Öğrenciler süre sınırı olmaksızın soruları cevaplamışlardır. Öğrencilerin testte yapmış oldukları doğru cevap sayıları ve test süreleri tutulmuştur. Daha sonra öğrencilerle görüşme yapılmıştır. Uzman görüşleri ve ön pilot uygulamasından sonra pilot uygulaması yapılacak 18 kod senaryosu belirlenmiştir (Tablo 1).

Tablo 1. HAPT'in Pilot Uygulamasında Yer Alan Kod Senaryoları ve İçerdiği Hatalar

Madde No	Kod Senaryosu	İçerdiği Hata Türü
1	Sınav Puanlarına Göre Dersten Geçme Durumunu Kontrol Etme	Çalışma Zamanı Hatası
2	Öğrenci Kaydı Yapma	Mantık Hatası
3	Merhaba C#	Derleme Zamanı Hatası
4	'Bir Daha Yaramazlık Yapmayacağım' Diye Yazdırma	Çalışma Zamanı Hatası
5	Toplama ve Çıkarma Yapma	Çalışma Zamanı Hatası
6	Bir Sayının 3'e Bölünme Durumunu Kontrol Etme	Derleme Zamanı Hatası
7	Milli Piyango Sonuçları Yazdırma	Çalışma Zamanı Hatası
8	Sınav Notlarına Göre Başarı Ortalaması Hesaplama	Mantık Hatası
9	Kullanıcı Adı ve Şifre İle Giriş Yapma	Mantık Hatası
10	Ebeveyn İsimleri İle Aile İlişkilerini Yazdırma	Derleme Zamanı Hatası
11	Askere Gitme Durumunu Sorgulama	Mantık Hatası
12	Girilen Sayının Karesini ve Küpünü Hesaplama	Çalışma Zamanı Hatası
13	Ekrana Adını ve Soyadını Yazdırma	Mantık Hatası
14	Girilen Açılar İle Üçgen Olma Durumunu Kontrol Etme	Derleme Zamanı Hatası
15	Yıllara Göre Yaş Hesaplama	Çalışma Zamanı Hatası
16	Sayı Tahmini Oyunu	Derleme Zamanı Hatası
17	Girilen Beş Sayı İçinden En Küçük Olanını Bulma	Derleme Zamanı Hatası
18	Girilen Sayıların Toplamını Yazdırma	Mantık Hatası

Tablo 1'de belirtilen ve madde analizi için hazırlanan 1 ile 18 satır aralığında değişen kod senaryolarının her biri belirlenen hata türlerinden bir tane hata içermektedir. Testin pilot uygulaması için test formu web sayfası olarak tasarlanmıştır. Tasarlanan test formunun ekran görüntüsü Şekil 2'de gösterilmiştir.



Şekil 2. Testin pilot uygulamasının örnek ekran görüntüsü

Testin pilot uygulaması ilgili okulların bilgisayar laboratuvarlarında gerçekleştirilmiştir. Bunun için ilk önce öğrenciler laboratuvarlara davet edilmiştir. Uygulamaya başlamadan önce öğrencilere test hakkında bilgilendirmeler yapılmış ve varsa soruları cevaplandırılmıştır. Öğrenciler kod senaryolarındaki hatalara yönelik soruları cevaplandırmış, cevapları veri tabanına kaydedilmiştir. Test sonrasında öğrencilere teşekkür edilmiş ve sınıflarına yönlendirilmiştir.

Çalışmanın veri toplama süreci, Hacettepe Üniversitesi Etik Komisyonu, 08.08.2018 tarihli 35853172-300 belge sayı numaralı izni ile gerçekleştirilmiştir.

2.3. Veri Analizi ve Yorumlanması

Kapsam geçerliğinin sağlanması amacıyla alınan uzman görüşleri arasındaki uyumluluk düzeyi Cohen'in Kappa Katsayısı ile incelenmiştir. Bu katsayı ikili değişken kullanılarak, iki kişi tarafından aynı şeyin değerlendirildiği durumlarda, aralarındaki uyumun belirlenmesi amacı ile kullanılmaktadır (Doğan & Doğan, 2014). Bu katsayıya ilişkin yorumlama kriterleri Tablo 2'de verilmiştir.

Tablo 2. Cohen'in Kappa Katsayısının Yorumlanması

Kappa Katsayısı	Yorum*	Kappa Katsayısı	Yorum**
.00 – .20	Kötü Uyum	< .00	Uyum Yok
.20 – .40	Adil Uyum	.00 – .39	Zayıf Bir Uyum
.40 – .60	Orta Düzeyde Uyum	.40 – .75	İyi bir Uyum
.60 – .80	Önemli Uyum	.76 – 1.00	Mükemmel Bir Uyum
.80 – 1.00	Neredeyse Mükemmel Uyum		

* Landis ve Koch (1977); ** Alpar (2014)

Pilot uygulamada öğrencilerin cevaplamış oldukları testin geçerliğini sağlamak için madde analizi yapılmıştır. Madde analizleri yapılırken "TAP: Test Analysis Program" kullanılmıştır (Brooks & Johanson, 2003). İlk olarak madde güçlük indeksleri belirlenmiş olup, maddelerin zorluk düzeyleri hakkında bilgi vermiştir. Bu indeks maddeyi doğru cevaplayan birey sayısının tüm gruba oranı ile belirlenmektedir (Kan, 2014). Bu indeks, maddenin iyi veya kötü olduğunu belirlemek için yeterli değildir (Kilmen, 2017). Madde analizi için belirlenen madde ayırt edicilik indeksi alt-üst grup yöntemi kullanılarak belirlenmiştir. Bu yöntem, teste gösterilen başarı sıralamasının alttan ve üstten tüm grubun %27'sini oluşturan öğrencilerin maddeye vermiş oldukları cevapların farklılaşma durumunu belirlemek üzerine kuruludur (Kan, 2014). Madde güçlük indekslerinin ile madde ayırt edicilik indekslerinin yorumlama kriterleri aşağıda belirtilmiştir (Tablo 3).

Tablo 3. Madde Güçlük İndekslerinin ve Madde Ayırt Edicilik İndekslerinin Yorumlanması

Madde Güçlük İndeksi*	Yorum
.00 – .19	Çok Zor
.20 – .39	Zor
.40 – .59	Orta
.60 – .79	Kolay
.80 – .100	Çok Kolay
Madde Ayırt Edicilik İndeksi**	**Yorum**
.00 – .19	Ayırt Edici Değildir. Testten Çıkartılmalıdır
.20 – .29	Maddenin İncelenmesi Gereklidir.
.30 – .39	Madde İyi Bir Ayırt Edicidir.
.40 ve üzeri	Madde Çok İyi Bir Ayırt Edicidir.

* Başol (Başol, 2015); Kline (1986)'dan akt. Kan (2014) ** Başol (2015); Kilmen (2017); Büyüköztürk ve diğerleri (2014); Baykul (2010)

Testin güvenirliğine ilişkin çıkarım yapmak için tek uygulamaya dayalı güvenirlik belirleme yöntemlerinden olan Kuder-Richardson 20 (KR-20) formülü uygulanmıştır. Bu formül 1-0 (doğru-yanlış) şeklinde puanlanan ölçme araçlarının güvenirliğinin belirlenmesi için uygundur (Baykul, 2010; Demirtaşlı, 2017). Bu formül, güvenirliği belirleme yöntemlerinden olan Cronbach Alpha katsayısı ile aynı sonucu vermektedir (Alpar, 2014). Ayrıca KR-20 indeksi, bir diğer güvenirliği belirleme yöntemlerinden olan iki yarı güvenirlik tahminlerinin ortalaması ile aynı sonucu vermektedir (Cronback, 1951'den akt. Başol, 2015). Bu sebeplerden dolayı testin güvenirliğinin incelenmesi için KR-20 formülünün uygulanması uygun görülmüştür. Elde edilen güvenirlik

indeksine yönelik yapılacak yorumlamalar Tablo 4'te belirtilmiştir. Bunun yanında testin güvenirliğine ilişkin fikir vermesi amacıyla maddelerin güvenirlik katsayısı belirlenmiştir. Bu katsayı maddelerin ayırt edicilik indeksleri ile standart sapmalarının çarpımları ile belirlenmektedir (Başol, 2015; Kan, 2014).

Tablo 4. KR-20 İndeksinin Yorumlanması

KR-20 İndeksi*	Yorum
.00 – .39	Test Güvenilir Değildir.
.40 – .59	Testin Güvenirliği Düşüktür.
.60 – .79	Test Oldukça Güvenilirdir.
.80 – 1.00	Test Yüksek Güvenirliğe Sahiptir.

* Alpar (2014)

Hata ayıklama performansı testinin pilot uygulaması ve maddelerine yönelik yapılan analiz yöntemleri belirtilmiştir. Bu yöntemler doğrultusunda belirlenen değerlere yapılacak yorumlamalar da yukarıda ifade edilmiştir. Yapılan uzman görüşleri, ön pilot uygulaması ve analizler doğrultusunda elde edilen bulgular aşağıda verilmiştir.

3. Bulgular

Hata ayıklama performansı testini geliştirmek amacıyla hataların ve kapsamlarının belirlenmiş, ilgili alanyazın ve materyal incelemesi sonucunda uzman görüşleri alınmış, ön pilot uygulaması yapılmış ve karar kılınan kod senaryoları ile pilot uygulaması yapılmıştır. Tüm bu yapılan görüşmeler ve analizler sonucunda elde edilen bulgular başlıklar halinde sunulmuştur.

3.1. Uzman görüşlerine ilişkin bulgular

MEB'e bağlı mesleki ve teknik liselerin Bilişim Teknolojileri bölümlerinde görev yapan iki uzmandan görüşleri istenmiştir. Uzmanlar, yönergesiyle birlikte sunulan 27 kod senaryosu değerlendirmişlerdir. Vermiş oldukları görüşlerinde bazı kod senaryolarının uzun olduğu, kodların biçimsel özelliklerinin değiştirilmesi gerektiği, bazı değişkenlerin türlerinin ve isimlerinin değiştirilmesi gerektiği ve bazı kod senaryolarını çözmek için farklı bilgiler bilmek gerektiğine (Ör; matematiksel işlem önceliği) yönelik dönütler sağlamışlardır. Uzmanlar derleme zamanı hatası olan kodlarda hata satırının vurgulanması gerektiğini ifade etmiştir. Çünkü kodlamada kullanılan derleme yazılımlarının bunu yaptığını belirtmişlerdir. Uzmanlar ayrıca bu testin bilgisayar aracılığıyla cevaplandırılması gerektiğini belirtmişlerdir. Buna göre 11 kod senaryosunda yapısal (ör. kod satırlarının azaltılması), 5 kod senaryosunda biçimsel değişiklikler (ör. hata satırının vurgulanması, kodların renklendirilmesi) yapılmıştır. 8 kod senaryosu tam anlaşılamadığından dolayı değişiklikler yapılmıştır. 4 kod senaryosunda ise değişiklik yapılmamıştır. 4 kod senaryosu testten çıkartılmış ve uygun görülen 2 kod senaryosu teste eklenmiştir. Uzman görüşleri sonrasında düzenlenen 25 kod senaryosu ön pilot uygulaması yapmak için hazırlanmıştır.

3.2. Ön pilot uygulamasına ilişkin bulgular

Ön pilot uygulaması yapılırken pilot uygulaması yapılacak okullardan birinde programlama temelleri dersinde farklı düzeylerde (yüksek, orta ve düşük) başarı gösteren 3 öğrenci belirlenmiştir. Hazırlanan 25 kod senaryosu ve içerdikleri hatalara ilişkin sorular yönergesiyle birlikte sunulmuştur. Öğrenciler sorulara cevap verirken süre sınırlandırması yapılmamıştır. Yüksek düzey başarı gösteren öğrenci 19 (84 dakika), orta düzey başarı gösteren öğrenci 9 (59 dakika) ve düşük düzey başarı gösteren öğrenci 4 (17 dakika) soruya doğru cevap vermişlerdir. Test sonrasında öğrencilerle görüşmeler yapılmıştır. Bu görüşmelerde testte bulunan kod

senaryolarının derslerde yaptıkları uygulamalara benzediğini, içeren hatalarla sıklıkla karşılaştıklarını ve derslerde yapılan değerlendirmelerde bu tür sorular sorulduğunu ifade etmişlerdir. Öğrencilerin bu testte verilmesi gereken süre konusunda farklı görüşleri olmuştur. Öğrencilerin testte geçirmiş oldukları süreler dikkate alınarak bir takım düzenlemelere gidilmiştir. Bu düzenlemeler doğrultusunda uzman görüşleri alınarak benzer kod senaryoları çıkartılarak testin 18 kod senaryosuna düşürülmesi ve pilot uygulaması için 40 dakika süre verilmesi kararlaştırılmıştır.

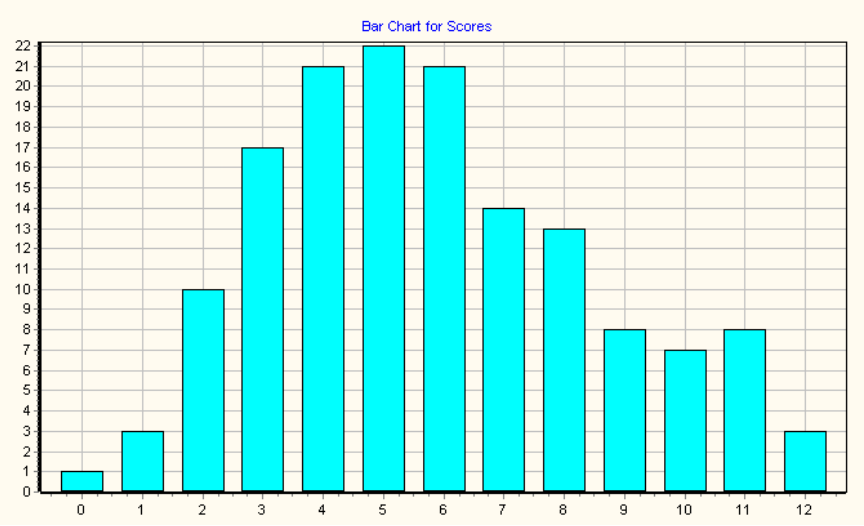
3.3. Pilot uygulamasına ilişkin bulgular

HAPT'in pilot uygulaması, Kastamonu ilinin farklı ilçelerinde bulunan beş farklı lisenin Bilişim Teknolojileri bölümlerinde okuyan 148 öğrenci ile birlikte gerçekleştirilmiştir. Uygulama okulların bilgisayar laboratuvarlarında yapılmıştır. Öğrenciler bireysel olarak bilgisayar başına geçmiş ve web tabanlı hazırlanan testi cevaplandırmışlardır. Sorulara doğru cevap verme 1, yanlış cevap verme veya boş bırakma durumunda ise 0 puan verilmiştir. Öğrencilerin cevapları doğrultusunda oluşan pilot uygulamaya yönelik test istatistikleri Tablo 5'te verilmiştir.

Tablo 5. HAPT'in Pilot Uygulamasına İlişkin Test İstatistikleri

Test İstatistikleri	Değer
Ortalama	5.82
Ortanca	5.50
Standart Sapma	2.70
Çarpıklık	.38
Basıklık	-.53
En Düşük	.00
En Yüksek	12.00

Tablo 5 incelendiğinde HAPT'in pilot uygulamasında verilen doğru cevapların ortalaması 5.82, ortancası 5.50'dir. Öğrencilerin verdikleri cevapların standart sapması 2.70 olmuştur. Testte en düşük 0 puan almışlarken, en yüksek 12 puan almışlardır. Testte çarpıklık değeri .38 olurken, basıklık değeri -.53 olmuştur. Çarpıklık ve basıklık değerlerine bakıldığında dağılımın normale yakın olduğu söylenebilir. Dağılıma ilişkin grafik aşağıda verilmiştir (Şekil 3).



Şekil 3. HAPT'in pilot uygulamasında verilen doğru cevapların dağılımı

Öğrencilerin verdikleri doğru cevapların dağılımları incelendiği zaman normale yakın bir dağılım gösterdiği görülmektedir. Ayrıca öğrencilerin büyük çoğunluğunun 3 ile 6 puan aralığında doğru cevap verdikleri (%55) görülmüştür (Şekil 3).

3.3.1. Testin geçerliğine ilişkin bulgular

3.3.1.1. Kapsam geçerliği

Testin kapsam geçerliğinin sağlanması adına uzman görüşlerine başvurulmuştur. Uzmanlar kod senaryolarının, içerdiği hataların ve hatalara yönelik sorulan soruların uygun olma durumlarına ilişkin görüşlerini belirtmişlerdir. Bu görüşleri uyumluluğu Cohen'in Kappa Katsayısı ile incelenmiştir. Yapılan analizde Cohen'in Kappa Katsayısı K=.76 olarak hesaplanmıştır. Buna göre uzmanların görüşleri arasındaki uyumun önemli veya mükemmel düzeyde olduğu söylenebilir (Alpar, 2014; Landis & Koch, 1977).

3.3.1.2. Madde güçlüğü

Testte yer alan maddelerin doğru cevaplanma durumuna ilişkin bir indekstir. Maddelerin zorluk durumlarına yönelik fikir vermektedir. Pilot uygulamasında maddelere ait güçlük indeksleri Tablo 6'da belirtilmiştir.

Tablo 6. HAPT'in Pilot Uygulamasında Madde Güçlük İndekslerine İlişkin Bulgular

Maddeler	Madde Güçlüğü (p_j)	Yorum	Maddeler	Madde Güçlüğü (p_j)	Yorum
1	.39	Zor	10	.28	Zor
2	.48	Orta Güçlükte	11	.39	Zor
3	.44	Orta Güçlükte	12	.19*	Çok Zor
4	.36	Zor	13	.18*	Çok Zor
5	.33	Zor	14	.32	Zor
6	.30	Zor	15	.42	Orta Güçlükte
7	.18*	Çok Zor	16	.31	Zor
8	.29	Zor	17	.18*	Çok Zor
9	.49	Orta Güçlükte	18	.31	Zor

* Madde güçlüğü bakımından çıkarılması uygun maddeler (Kline (1986)'dan akt. Kan, 2014)

Tablo 6'da görüldüğü üzere pilot uygulaması yapılan maddelerin güçlük indekslerinin .18 ile .49 arasında değişmektedir. Test maddelerinin çoğunun (10 madde) zor olarak nitelendirilebilecek düzeyde olduğu görülmüştür. Testte yer alan 4 maddenin orta düzeyde olduğu belirlenmiştir. Testte yer alan 4 maddenin de çok zor düzeyde olduğu ve testten çıkarılması gerektiği görülmüştür.

3.3.1.3. Madde ayırt ediciliği

Madde güçlük indeksi madde ile belirlenmek istenen özelliğe sahip olan ile olmayan öğrencileri ayırabilmesine olanak sağlayan, iyi ve kaliteli maddelerin belirlenmesinde kullanılan bir indekstir. Testin pilot uygulamasındaki maddelere ait ayırt edicilik indeksleri Tablo 7'de sunulmuştur.

Tablo 7. HAPT'in Pilot Uygulamasında Madde Ayırt Edicilik İndekslerine İlişkin Bulgular

Maddeler	Alt Grupta Doğru Sayısı	Üst Grupta Doğru Sayısı	Madde Ayırt Ediciliği (r_{jx})	Yorum
1	7	34	.51	Çok İyi
2	18	35	.31	İyi
3	8	37	.54	Çok İyi
4	5	34	.55	Çok İyi
5	11	24	.24	Düzeltilmeli
6	7	26	.36	İyi
7	6	15	.17*	Testten Çıkarılmalı
8	9	24	.28	Düzeltilmeli
9	18	35	.31	İyi
10	9	16	.13*	Testten Çıkarılmalı
11	13	31	.33	İyi
12	6	14	.15*	Testten Çıkarılmalı
13	3	13	.19*	Testten Çıkarılmalı
14	4	31	.51	Çok İyi
15	8	38	.56	Çok İyi
16	11	21	.18*	Testten Çıkarılmalı
17	5	11	.11*	Testten Çıkarılmalı
18	10	29	.35	İyi

* Madde ayırt ediciliği bakımından testten çıkarılması gereken maddeler (Başol, 2015)

Tablo 7 incelendiğinde test maddelerinin madde ayırt edicilik indekslerinin .11 ile .56 arasında değiştiği görülmüştür. Buna göre testteki 5 maddenin çok iyi düzeyde ve 5 maddenin iyi düzeyde ayırt edici olduğu görülmüştür. 2 maddenin (5. ve 8. maddeler) düzenlenmesi ve gerekli görülürse çıkarılması gerektiği belirlenmiştir. Testte yer alan 7. 10. 12. 13. 16. ve 17. maddelerin ayırt ediciliklerinin yeterli düzeyde olmadığı ve testte çıkarılması gerektiği belirlenmiştir.

3.3.2. Testin güvenirliğine ilişkin bulgular

3.3.2.1. KR-20 indeksi

HAPT'in güvenirliğinin incelenmesi için KR-20 indeksi belirlenmiştir. Bu indeks maddelerin birbirleri ile ne düzeyde tutarlı olduğunu göstermekte, bir diğer ifade ile iç tutarlığını belirtmektedir. Teste ilişkin hesaplanan KR-20 indeksi Tablo 8'de verilmiştir.

Tablo 8. HAPT'in Pilot Uygulamasına Yönelik Hesaplanan KR-20 İndeksi

Madde Sayısı	KR-20 İndeksi
18	.514

Yapılan hesaplamalar sonucunda KR-20 indeksinin, HAPT'in düşük düzey bir güvenirliğe sahip olduğunu göstermiştir (Tablo 8). Testin güvenirliğini artırmak için çıkarılması gereken maddeler incelenmiştir (Tablo 9).

Tablo 9. HAPT'in Güvenirliğini Artırmak İçin Çıkarılması Gereken Maddelere İlişkin Bulgular

Maddeler	Madde Silindiğinde Ortalama	Madde Silindiğinde Standart Sapma	Madde Silindiğinde KR-20 İndeksi
1	5.43	2.52	.474
2	5.34	2.60	.510
3	5.38	2.50	.467
4	5.46	2.48	.451
5	5.49	2.62	.515*
6	5.52	2.60	.504
7	5.64	2.64	.512
8	5.53	2.61	.510
9	5.33	2.58	.502
10	5.54	2.66	.528*
11	5.43	2.55	.487
12	5.63	2.65	.515*
13	5.64	2.64	.511
14	5.50	2.52	.470
15	5.40	2.49	.462
16	5.51	2.69	.541*
17	5.65	2.68	.529*
18	5.51	2.53	.476

* HAPT güvenirliğinin artırılması için testten çıkarılması önerilen maddeler

Tablo 9'da maddelerin testten çıkarılması halinde KR-20 indeksinin nasıl değişeceği gösterilmektedir. Buna göre 5., 10., 12., 16. ve 17. maddelerin testten çıkarılması halinde testin iç tutarlılığının artacağı belirlenmiştir.

3.3.2.2. Madde güvenirliği

Madde güvenirliği, testin güvenirliği ile doğrudan ilişkili olup (Kan, 2014), testin güvenirlik katsayısının artırılması amacıyla incelenmesi gereklidir. HAPT'in pilot uygulamasında yer alan maddelerin güvenirlikleri Tablo 10'da belirtilmiştir.

Tablo 10. HAPT'in Pilot Uygulamasında Yer Alan Maddelerin Güvenirlik İndekslerine İlişkin Bulgular

Maddeler	Madde Güvenirliği (r_j)	Maddeler	Madde Güvenirliği (r_j)
1	.25	10	.06*
2	.16	11	.16
3	.27	12	.06*
4	.26	13	.07*
5	.11	14	.24
6	.17	15	.28
7	.07*	16	.08*
8	.13	17	.04*
9	.16	18	.16

* HAPT'te bulunan güvenirlik indeksi düşük maddeler

HAPT'in pilot uygulamasında incelenen maddelerinin güvenirlik indeksleri Tablo 10'da belirtilmiştir. Buna göre 7., 10., 12., 13., 16. ve 17. maddelerin madde güvenirlik indekslerinin düşük olduğu ve testten çıkarılmasıyla test güvenirliğinin artacağı ifade edilebilir.

3.3.3. Maddelerin seçimi ve testin son haline ilişkin bulgular

Pilot uygulama sonrasında yapılan analizler sonucunda 5., 7., 10., 12., 13., 16. ve 17. maddelerin testten çıkarılması kararlaştırılmıştır. Madde ayırt ediciliği bakımından düzenlenmesi gereken 8. madde ise uzman görüşü alınarak testten çıkarılmıştır. Sonuç olarak 3 derleme zamanı hatası, 3 çalışma zamanı hatası ve 4 mantık hatası içeren 10 kod senaryosundan ve hatalara yönelik sorular

içeren HAPT belirlenmiştir. HAPT'in nihai halinde bulunan maddelere ait güçlük indeksleri, ayırt edicilik indeksleri ve güvenirlik indeksleri Tablo 11'de verilmiştir.

Tablo 11. HAPT'in Nihai Halinin Madde Analizine İlişkin Bulgular

Madde	p_j	r_{jx}	r_j	Madde	p_j	r_{jx}	r_j
1	.39	.45	.22	10	.49	.47	.24
2	.48	.44	.22	11	.39	.41	.20
3	.45	.56	.28	14	.32	.49	.23
4	.36	.69	.33	15	.43	.55	.27
9	.30	.39	.18	18	.31	.46	.21

Tablo 11 incelendiğinde maddelerin güçlük indekslerinin .30 ile .49 arasında, madde ayırt edicilik indekslerinin .39 ile .69 arasında değiştiği görülmektedir. Madde güvenirlik indekslerinin ise .20 ile .33 arasında değişmektedir. Testin madde güçlük indekslerinin, madde ayırt edicilik indekslerinin ve madde güvenirlik indekslerinin uygun olduğu görülmüştür. HAPT'in nihai halinin güvenirliğini belirlemek için hesaplanan KR-20 indeksine ilişkin bulgular Tablo 12'de belirtilmiştir.

Tablo 12. HAPT'in Nihai Haline Yönelik Hesaplanan KR-20 İndeksi

Madde Sayısı	KR-20 İndeksi
10	.575

HAPT'in nihai haline yönelik hesaplanan KR-20 indeksi incelendiğinde testin güvenirlik düzeyini yükseldiği görülmüştür (Tablo 12).

4. Tartışma ve Sonuç

Bu çalışma kapsamında mesleki ve teknik lisesi Bilişim Teknolojileri bölümü öğrencilerinin hata ayıklama performanslarını değerlendirmek için bir başarı testinin geliştirilmesi amaçlanmıştır. Bu amaçla bazı adımlar izlenmiştir. Testin amacı, karşılaşılan hatalar ve kapsamları belirlenmiş, alanyazında yer alan testler incelenmiştir. Uzman görüşleri alınmış, ön pilot uygulaması yapılarak testin pilot uygulamasında yer alacak kod senaryoları belirlenmiştir. Testin pilot uygulaması sonrasında öğrencilerin teste vermiş oldukları cevapları göz önünde bulundurularak madde analizi ve güvenirlik analizi yapılmıştır. Uygun görülen maddeler çıkarılmış ve testin son hali oluşturulmuştur.

Test, 10 kod senaryosu, her kod senaryosunda birer hata ve bu hatalara yönelik sorulardan oluşmaktadır. Oluşturulan teste ait maddelerin analizleri yapılmıştır. Buna göre testte yer alan maddelerin ayırt edicilik indekslerinin büyük bir çoğunluğu .40'ın üzerindedir. Buna göre maddelerin büyük bir çoğunluğunun çok iyi ayırt edici olduğu görülmüştür. Sadece bir maddenin ayırt edicilik indeksi .39 olup bu maddenin de iyi bir ayırt edici olduğu söylenebilir. Testin ortalama ayırt edicilik indeksi ise .49 olarak belirlenmiştir. Dolayısıyla testin geneli itibariyle ayırt edicilik gücünün çok iyi olduğu yorumu yapılabilir (Başol, 2015; Baykul, 2010; Büyüköztürk ve diğerleri, 2014; Kilmen, 2017). Testin maddelerine yönelik yapılan analizlerden bir diğeri madde güçlük indeksinin hesaplanmasıdır. Buna göre test maddelerinin yaklaşık yarısının .40'ın altında olduğu, geriye kalan maddelerin ise üzerinde olduğu görülmüştür. Buna göre testin yarısının zor maddelerden, yarısının da orta güçlükte maddelerden oluştuğu yorumu yapılabilir. Testin ortalama güçlüğü incelendiğinde ise .39'dur. Buna göre testin yaklaşık olarak orta düzeyde bir güçlüğü olduğu söylenebilir (Başol, 2015; Kline (1986)'dan akt. Kan, 2014). Testin güvenirliğine ilişkin belirlenen KR-20 indeksinin .575 olduğu görülmüştür. Bu değer, testin düşük düzey güvenirliğe sahip olduğunu gösterse de (Alpar, 2014), testin maddelerinin yarısının güçlük

düzeylerinin zor olması (Başol, 2015) ve testte bulunan madde sayılarının az olması (Alpar, 2014; Baykul, 2010; Özbek, 2014; Sönmez & Alacapınar, 2016) bu değeri etkilemektedir. Buna göre testte bulunan madde sayılarının 10-15 aralığında olması halinde .50 güvenirlik indeksinin yeterli olduğu ve testin güvenilir olduğu yorumu yapılabilmektedir (Alpar, 2014).

Sonuç olarak, MEB'e bağlı mesleki ve teknik liseleri Bilişim Teknolojileri bölümü öğrencilerinin programlama süreçlerinde karşılaştıkları hataları ayıklama performanslarını değerlendirmek amacıyla 10 kod senaryosundan oluşan test geliştirilmiştir. Bu teste yönelik yapılan analizler doğrultusunda testin geçerli ve güvenilir değerlere sahip olduğu ifade edilebilir.

MEB'e bağlı okul ve kurumlarda yapılacak araştırmalarda MEB tarafından uygulanan yönergede "Araştırmanın veri toplama araçlarını katılımcılara uygulanma süresi, bir ders saatinden fazla olamaz." maddesi bulunmaktadır (Milli Eğitim Bakanlığı, 2007). Dolayısıyla testin pilot uygulaması bir ders saati (40 dakika) ile sınırlandırılmış, öğrencilerin bu süre zarfında cevaplandıracakları kod senaryoları sayısı buna göre belirlenmiştir. Ayrıca MEB tarafından verilen mesleki ve teknik liselerinin programlama eğitiminde C# programlama dili kullanıldığından dolayı testi oluşturan kod senaryoları C# programlama dili ile geliştirilmiştir. Dolayısıyla literatürde farklı programlama dili ve düzeylerinde hata ayıklama testlerine ihtiyaç bulunmaktadır. Bu bağlamda araştırmacılara farklı dillerde ve düzeylerde hata ayıklama performansı testleri geliştirmeleri önerilmektedir.

Teşekkür

Bu çalışma hazırlanması sürecinde TÜBİTAK tarafından "2211-A Genel Yurt İçi Doktora Burs Programı" kapsamında verilen destek için teşekkür ederiz.

Kaynakça

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, 37*(3), 84–88. https://doi.org/10.1145/1151954.1067472

Albert. (2018). *AP® Computer Science A study guide*. Retrieved from https://www.albert.io/ap-computer-science-a.

Alpar, R. (2014). *Spor, sağlık ve eğitim bilimlerinden örneklerle uygulamalı istatistik ve geçerlik-güvenirlik (3rd ed.)*. Ankara: Detay Yayıncılık.

Altun, A., & Kasalak, İ. (2018). Blok temelli programlamaya ilişkin öz-yeterlik algısı ölçeği geliştirme çalışması: Scratch örneği. *Eğitim Teknolojisi Kuram ve Uygulama, 8*(1), 209–225.

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin öz yeterlilik algısı ölçeğinin Türkçe formumun geçerlilik ve güvenirlik çalışması. *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 3*(2), 297–308.

Başol, G. (2015). *Eğitimde ölçme ve değerlendirme.* Ankara: Pegem Akademi Yayıncılık.

Baykul, Y. (2010). *Eğitimde ve psikolojide ölçme: Klasik test teorisi ve uygulaması.* Ankara: Pegem Akademi Yayıncılık.

Bednarik, R. (2012). Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human Computer Studies, 70*(2), 143–155. https://doi.org/10.1016/j.ijhcs.2011.09.003

Bednarik, R., & Tukiainen, M. (2004a). Visual attention and representation switching in Java program debugging: A study using eye movement tracking. In E. Dunican & T. R. G. Green (Eds.), *Proceedings of the 16th Workshop of the Psychology of Programming Interest Group* (pp. 159–169). https://doi.org/http://doi.acm.org/10.1145/1028014.1028066

Bednarik, R., & Tukiainen, M. (2004b). Visual attention tracking during program debugging. *Proceedings of the 3rd Nordic Conference on Human-Computer Interaction*, 331–334. https://doi.org/10.1145/1028014.1028066

Bednarik, R., & Tukiainen, M. (2008). Temporal eye-tracking data: Evolution of debugging strategies with multiple representations. *Proceedings of the Eye Tracking Research & Application Symposium, ETRA 2008*, 99–102. https://doi.org/10.1145/1344471.1344497

Brooks, G. P., & Johanson, G. A. (2003). TAP: Test analysis program. *Applied Psychological Measurement, 27*(4), 303–304. https://doi.org/10.1177/0146621603027004007

Büyüköztürk, S., Çakmak Kılıç, E., Akgün, Ö. E., Karadeniz, Ş., & Demirel, F. (2014). *Bilimsel araştırma yöntemleri*. Ankara: Pegem Akademi Yayıncılık.

Chen, M., & Lim, V. (2013). Eye gaze and mouse cursor relationship in a debugging task. In C. Stephanidis (Ed.), *Communications in Computer and Information Science* (Vol. 373, pp. 468–472). https://doi.org/10.1007/978-3-642-39473-7_93

Chen, M. W., Wu, C. C., & Lin, Y. T. (2013). Novices' debugging behaviors in VB programming. *2013 Learning and Teaching in Computing and Engineering*, 25–30. https://doi.org/10.1109/LaTiCE.2013.38

Cronback, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334. https://doi.org/10.1007/BF02310555

Demirtaşlı, R. N. (2017). Ölçmede güvenirlik. İçinde R. N. Demirtaşlı (Ed.), *Eğitimde Ölçme ve Değerlendirme (4. Baskı, ss. 77–100).* Ankara: Anı Yayıncılık.

Doğan, İ., & Doğan, N. (2014). *Adım adım çözümlü parametrik olmayan istatistiksel yöntemler*. Ankara: Detay Yayıncılık.

Dooley, J. (2011). Debugging. *In Software Development and Professional Practice* (pp. 181–192). New York: Springer.

Downey, A. B., & Mayfield, C. (2016). Think Java: How to think like a computer scientist. USA, Massachusetts: Green Tea Press

Ducasse, M., & Emde, A.-M. (1988). A review of automated debugging systems: Knowledge, strategies and techniques. *Proceedings of the 10th International Conference on Software Engineering*, 162–171. Singapore: IEEE.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education, 18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Grigoreanu, V., Brundage, J., Bahna, E., Burnett, M., Elrif, P., & Snover, J. (2009). Males' and females' script debugging strategies. In V. Pipek, M. B. Rosson, B. de Ruyter, & V. Wulf (Eds.), *End-User Development* (Vol. 5435, pp. 205–224). https://doi.org/10.1007/978-3-642-00427-8_12

Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and correcting Java programming errors for introductory computer science students. In S. Grissom, D. Knox, D. T. Joyce, & W. Dann (Eds.), *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)* (pp. 153–156). Reno, Nevada: Association for Computing Machinery.

Institute of Electrical and Electronics Engineers. (2010). *IEEE standard classification for software anomalies*. New York.

Johnson, W. L., Soloway, E., Cutler, B., & Draper, S. (1983). *Bug catalogue: I*. New Haven, Connecticut.

Kan, A. (2014). Ölçme aracı geliştirme. İçinde S. Tekindal (Ed.), *Eğitimde Ölçme ve Değerlendirme (4. Baskı ss. 259–296)*. Ankara: Pegem Akademi Yayıncılık.

Karakoç, F. Y., & Dönmez, L. (2014). Ölçek geliştirme çalışmalarında temel ilkeler. *Tıp Eğitimi Dünyası, 40*, 39–48.

Khan, I. A., Brinkman, W. P., & Hierons, R. M. (2011). Do moods affect programmers' debug performance? *Cognition, Technology and Work, 13*(4), 245–258. https://doi.org/10.1007/s10111-010-0164-1

Kilmen, S. (2017). Madde analizi, madde seçimi ve yorumlanması. İçinde R. N. Demirtaşlı (Ed.), *Eğitimde Ölçme ve Değerlendirme (4. Baskı, ss. 327–348)*. Ankara: Anı Yayıncılık.

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*, 159–174.

Liu, Z., Zhi, R., Hicks, A., & Barnes, T. (2017). Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education, 27*(1), 1–29. https://doi.org/10.1080/08993408.2017.1308651

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education, 18*(2), 67–92. https://doi.org/10.1080/08993400802114581

Milli Eğitim Bakanlığı. (2007). *Milli eğitim bakanlığına bağlı okul ve kurumlarda yapılacak araştırma ve araştırma desteğine yönelik izin ve uygulama yönergesi*. Ankara. http://mevzuat.meb.gov.tr/dosyalar/538.pdf adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011a). *Basit kodlar.* Ankara. http://www.megep.meb.gov.tr/mte_program_modul/moduller/Basit%20Kodlar.pdf adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011b). *Bilişim teknolojileri alanı çerçeve öğretim programı.* Ankara. http://www.megep.meb.gov.tr/?page=ogretimProgramlari adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011c). *Kodlamaya hazırlık.* Ankara. http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Kodlamaya%20Haz%C4%B1rl%C4%B1k.pdf adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011d). *Kontrol deyimleri.* Ankara. http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Kontrol%20Deyimleri.pdf adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011e). *Metotlar.* Ankara. http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/metotlar.pdf adresinden erişilmiştir.

Milli Eğitim Bakanlığı. (2011f). *Programlama temelleri ders bilgi formu.* Ankara. http://www.megep.meb.gov.tr/dokumanlar/Ders%20Bilgi%20Formlar%C4%B1/B%C4%B0L%C4%B0%C5%9E%C4%B0M%20TEKNOLOJ%C4%B0LER%C4%B0_DBF_10.rar adresinden erişilmiştir.

Özbek, Ö. Y. (2014). Ölçme araçlarında bulunması istenen nitelikler. İçinde S. Tekindal (Ed.), *Eğitimde Ölçme ve Değerlendirme (4. Baskı., ss. 43–91).* Ankara: Pegem Akademi Yayıncılık.

Peng, F., Li, C., Song, X., Hu, W., & Feng, G. (2016). An eye tracking research on debugging strategies towards different types of bugs. *2016 IEEE 40th Annual Computer Software and Applications Conference,* 130–134. https://doi.org/10.1109/COMPSAC.2016.57

Romero, P., Cox, R., du Boulay, B., & Lutz, R. (2002). Visual attention and representation Switching during java program debugging: A study using the restricted focus viewer. In M. Hegarty, B. Meyer, & N. H. Narayanan (Eds.), *Diagrammatic Representation and Inference* (pp. 221–235).

Romero, P., du Boulay, B., Cox, R., Lutz, R., & Bryant, S. (2007). Debugging strategies and tactics in a multi-representation software environment. *International Journal of Human Computer Studies, 65*(12), 992–1009. https://doi.org/10.1016/j.ijhcs.2007.07.005

Romero, P., Lutz, R., Cox, R., & du Boulay, B. (2002). Co-ordination of multiple external representations during Java program debugging. *Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, 207–214. https://doi.org/10.1109/HCC.2002.1046373

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering, SE-10*(5), 595–609.

Sönmez, V., & Alacapınar, F. G. (2016). *Sosyal bilimlerde ölçme aracı hazırlama*. Ankara: Anı.

Spohrer, J. C., Soloway, E., & Pope, E. (1985). A goal/plan analysis of buggy pascal programs. *Human-Computer Interaction, 1*, 163–207. https://doi.org/10.1207/s15327051hci0102_4

Yen, C. Z., Wu, P. H., & Lin, C. F. (2012). Analysis of experts' and novices' thinking process in program debugging. In K. C. Li, F. L. Wang, K. S. Yuen, S. K. S. Cheung, & R. Kwan (Eds.), *Engaging Learners Through Emerging Technologies* (pp. 122–134). https://doi.org/10.1007/978-3-642-31398-1_12

Yılmaz, F. (2013). *Meslek yüksek okulu öğrencilerinin programlama başarisini etkileyen faktörlerin incelenmesi.* Yüksek Lisans Tezi. Gazi Üniversitesi, Eğitim Bilimleri Enstitüsü. Ankara.

Extended Summary

1. Introduction

Programming is to produce computer-based solutions for a problem by situation analysis, algorithm and program design, implementation, testing and debugging. It is a usual situation that encounter bugs in the programming process. The process of resolving the bugs and making the generated program codes free of bugs is called debugging (Dooley, 2011).

Bugs are classified to find and to facilitate the solution processes. Although many classifications of bugs are made (Ahmadzadeh, Elliman, & Higgins, 2005; Downey & Mayfield, 2016; Hristova, Misra, Rutter, & Mercuri, 2003; Institute of Electrical and Electronics Engineers, 2010; Johnson, Soloway, Cutler, & Draper, 1983), these classifications are generally divided into three. Bugs types are compile-time bugs that are about characteristics of a programming language; run-time bugs that are about unexpected situations with misinterpretation of commands like an endless loop; and logical bugs that are about caused by the programmer's misconceptions, which do not provide output for the purpose.

The programming has an important place in the Information Technology department of vocational and technical high schools in Turkey. IT students take various programming courses during their high school education (Milli Eğitim Bakanlığı, 2011b). It is aimed to gain knowledge and skills like to know about bugs; check for bugs; understand bugs messages; detect and solution bugs in these courses (Milli Eğitim Bakanlığı, 2011f). In the literature, students who have this knowledge and skills are considered as good debuggers, while students who do not have these qualifications as poor debuggers (Ahmadzadeh et al., 2005). There are several approaches to conducting this assessment and to determine students' debugging performance.

In various studies, students debugged different types. But, In the literature, some instruments used are inadequate and not prepared for measuring debugging performance. Also, in these studies included participants with different characteristics. In addition, some studies have not provided validity and reliability information for debugging tasks. Furthermore, Java programming language is used for debugging tasks in the majority of studies. Therefore, in the literature, there is a lack of suitable measuring instruments for determining the debugging performance of students at the Information Technology department of vocational and technical high schools in Turkey. In this study, it is aimed to develop an achievement test to evaluate the debugging performance of students studying in the Information Technologies departments of vocational and technical high schools in Turkey.

2. Method

Firstly, bugs and their scope were determined to develop the Debugging Performance Test. Other exams and materials in the literature were examined. Then, 27 code scenarios with different types of bugs (9 compile-time bugs, 9 run-time bugs, and 9 logical bugs) were created and expert opinion was obtained. A pilot implementation of the test was performed. Following the pilot test, items were analyzed. After, item analysis, the test was finalized.

Expert opinions were obtained from two IT teachers for the test. Three IT department students participated in the pre-pilot implementation. In additon, 148 students who study the IT department of vocational and technical high schools participated in the pilot implementation of the test. In this process, the data were collected with paper-pen or web-based test environment.

In the analysis of the data, the compatibility of the expert opinions was examined with Cohen’s Kappa Coefficient. “TAP (Test Analysis Program)” was used for item analysis (Brooks & Johanson, 2003). Difficulty and discrimination indexes of the items were examined. In addition, the KR-20

index was examined for determining test reliability. Also, the reliability coefficients of the items were examined.

3. Findings

Compliance of experts to ensure content validity is significant level (Alpar, 2014; Landis & Koch, 1977). Afterward that expert opinions and pre-pilot implementation, 18 code scenarios were determined for pilot implementation. When item difficulty indexes and item discrimination indexes were evaluated, items 7, 10, 12, 13, 16 and 17 were removed. Also, item 5 was removed because of internal consistency index of the test will increase. In addition, item 8, which had to be improved in terms of item discrimination, was excluded from the test after obtaining expert opinions.

4. Results

The 10-item Debugging Performance Test was prepared by considering the findings obtained from the analyzes. Items in the test included 3 compile-time bugs, 3 run-time bugs, and 4 logical bugs. The item difficulty indexes are between .30 and .49, the item discrimination indexes are between .39 and .69, and the item reliability indexes are between .20 and .33 in the test. Also, the KR-20 reliability index of the test is .575. Therefore, it can be said that The Debugging Performance Test is valid and reliable.

Etik Beyannamesi
Bu makalede "Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesi" kapsamında belirtilen bütün kurallara uyduğumuzu, "Bilimsel Araştırma ve Yayın Etiğine Aykırı Eylemler" başlığı altında belirtilen eylemlerden hiçbirini gerçekleştirmediğimizi, hiçbir çıkar çatışmasının olmadığını ve oluşabilecek her türlü etik ihlalinde sorumluluğun makale yazarlarına ait olduğunu beyan ederiz.

Etik Kurul İzin Bilgileri
Etik kurul adı: Hacettepe Üniversitesi Etik Komisyonu
Etik kurul karar tarihi: 08.08.2018
Etik kurul belgesi sayı numarası: 35853172-300

Araştırma makalesi: Akçay, A., & Altun, A. (2021). Hata ayıklama performansı testi geliştirme: Geçerlik ve güvenirlik çalışması. *Erzincan Üniversitesi Eğitim Fakültesi Dergisi, 23*(3), 667-685.