Xamarin Test Bulutu üzerinde Mobil Uygulama Testi

Akhan Akbulut, Çağatay Çatal, Ece Demiray, Okan Özen
İstanbul Kültür Üniversitesi, Bilgisayar Mühendisliği Bölümü
{a.akbulut, c.catal}@iku.edu.tr
{1201020603, 1201020609}@stu.iku.edu.tr

Özet. Mobil cihaz kullanımındaki artışa paralel olarak yaygınlaşmaya başlayan mobil uygulamalar, kullanım alanı zenginliği ve platform çeşitliliği yönünden son kullanıcı ihtiyaçlarına önemli çözümler sağlamaktadır. iOS veya Android platform kullanımı, cihazların fiziksel özelliklerine bağımlılığı ortadan kaldırmakta ve uygulamaları; işlemci, RAM, ekran boyutu ve işletim sistemi sürümü farklılığı gibi özelliklere bakılmaksızın ürün çalışabilirliğini tüm uyumlu cihazlar üzerinde etkin kılmaktadır. Uygulamaların desteklediği ortamların artması, çeşitlilik yelpazesini olumlu yönde genişletmekte iken; çok farklı sayıda konfigurasyon için de uygulama testlerinin yapılması zorunluluğunu beraberinde getirmektedir. Mobil uygulama testi alanının, temel araştırma konularında, ilk sırada mobil cihaz çeşitliliği ve mobil platformların değişkenliği sorunu yer almaktadır. Bu çalışmada, Xamarin test bulut (Test Cloud) ortamı kullanılarak, sınırsız sayıda konfigurasyon için arayüz testlerinin otomasyonu sağlanmıştır. Çalışma kapsamında kullanılacak olan Xamarin test bulutu, Calabash çerçevesini kullanarak Android ve iOS mobil platformları için makine kodlu (native) veya melez uygulamaları test edebilmektedir. C# ve Ruby dili ile geliştiricilerin, birim test yazmaları gereken bu ortam için Andoid apk dosyalarının bulut üzerinde, arayüz kontrolleri için hazırladığımız şablon test durumları referans alınarak mobil uygulamaları otomatik olarak test eden bir uygulama geliştirilmiştir. 1000 adet farklı konfigurasyona sahip ortam için aynı test gerçeklenebilirken, sanallaştırma teknolojisi; erişim kolaylığı, test kapsamının genişlemesi, test maliyetlerini azaltması, zamandan tasarruf edilmesi ve çevik yazılım geliştirmeyi desteklemesi açısından önemli katkılar sunmaktadır. Üreticilerin Android işletim sistemlerini özelleştirdikleri düşünüldüğünde, test otomasyonunun sağlandığı bu uygulamayla birlikte, test uzmanları kaynak kod yazımını tekrarlı işler için gerçekleştirmemekte, hazırladığımız şablon test durumlarından yararlanmaktadır.

Anahtar Kelimeler: Mobil Uygulama Testi, Xamarin Test Cloud, Arayüz Testi, Test Otomasyonu, Mobil Uygulama, Bulut Bilişim

1 GİRİŞ

Günümüzde hemen hemen her kullanıcının beklenildiğinden farklı şekilde çalışan ve istenmeyen sonuçlar üreten bir uygulama ile kullanım tecrübesi bulunmaktadır. Bu tarz uygulamaların bireyler ve işletmeler üzerinde çok çeşitli olumsuzluklara neden olduk-

ları bilinmektedir. Bunların başında; vakit, para, itibar azalması, hatta sistemin kritikliğine bağlı olarak hayat kaybı bile yaşanabilmektedir. Uygulamaların etkinliğinin yayınlanmalarından önce çeşitli test faaliyetleri ile sınanması ve doğrulanması çok büyük önem taşımaktadır.

ANSI/IEEE 1059 uluslararası standardına göre test işlemi; bir yazılım öğesinin mevcut ve gerekli koşulları arasındaki farklılıklarını (kusur / sorun / hata) tespit etmek ve yazılım öğesinin özelliklerini değerlendirmek olarak ifade edilmektedir. İşletmelerde bu alandaki çalışmaları icra etmek üzere, çalışanların bilgi ve tecrübe seviyesine göre, test uzmanı, yazılım kalite güvence mühendisi veya kalite güvence analisti gibi farklı unvanlarda çalışan test personelleri görev almaktadır. Test faaliyetleri esnasında test otomasyonu yapabilen yardımcı yazılımlardan faydalanılması, süre ve iş gücü yönünden olumlu etkiler bırakmaktadır.

Günümüzde giderek yaygınlaşan mobil cihaz kullanımı, mobil uygulamaların geliştirmesini arttırmakta ve etkin mobil uygulamaların belirlenebilmesi için geleneksel yaklaşımlardan farklı test yöntemlerinin kullanılmasını gerektirmektedir. Mobil test işlemleri için en çok tercih edilen çerçeveler; Appium [1], Calabash [2], Robotium [3], Espresso [4], UI Automator [5] ve MobSF [6] olarak özetlenebilir.

Bu çalışma kapsamında Calabash çerçevesini kullanarak arayüz testlerinin gerçekleştiren Xamarin test bulut (Test Cloud) [7] ortamı ile arayüz testlerinin otomasyonuna dair bir örnek verilecektir. İkinci bölümde Mobil test otomasyon faaliyetlerine ilişkin çalışmalar anlatılmış olup, üçüncü bölümde geliştirilen uygulamanın yöntem detayları sunulmaktadır. Dördüncü bölümde ise, örnek bir uygulama üzerinde gerçekleştirilen test sonuçları verilmektedir.

2 İLİŞKİLİ ÇALIŞMALAR

Cambridge Üniversitesi'nin 2013 yılında yaptığı bir çalışmaya göre, küresel düzeyde yazılım hatalarının bulunma ve yok edilme maliyeti yıllık olarak 312 milyar dolardır [8]. Bu yüksek maliyetin azaltılması için test konusunda çalışan araştırmacılar, test otomasyonu alanına ağırlık vermeye başlamıştır. Test faaliyetlerinin yeterli olmadığı durumlarda, yazılım geliştirme maliyetlerinin, bakım aşamasından kaynaklı olarak çok fazla arttığı bilinmektedir. Test uzmanını merkez alan manuel test yöntemleri yerine, otomatikleştirilmiş test kullanımı zaman ve çaba açısından önemli tasarruflar sağlamaktadır. Bu nedenlerle; test otomasyonu, karmaşıklaşan yazılım sistemleriyle birlikte daha fazla önem kazanmış bir alan olarak karşımızda durmaktadır. Manuel testlerin her durumda tamamen otomatikleştirilmiş hale getirilmesi uygulanabilir bir yöntem gibi görünse de, bu çoğu durumda ekonomik açıdan akılcı bir yöntem değildir.

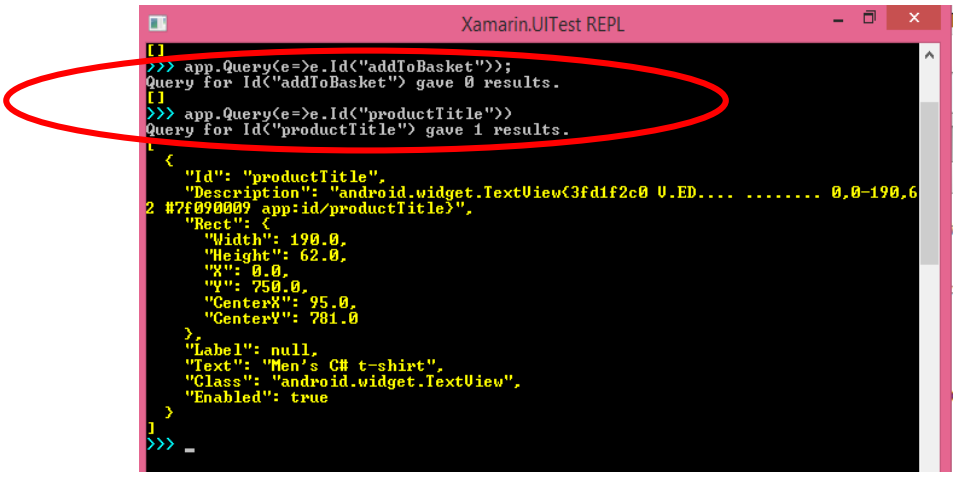
Yazılım testlerinin ne zaman otomatikleştirilmesi ve nelerin otomatikleştirilmesi gerektiği konusunda, Garousi ve Mantyla (2016), 78 kaynağı inceleyerek Çoksesli Literatür Taraması (Multivocal Literature Review-MLR) çalışmasını gerçekleştirmiştir [9].

26 kaynak akademik kaynaklardan alınırken, 52 kaynak gri literatür dediğimiz yayınlanmamış ve araştırma çalışması olmayan kaynaklardan (blog, tanıtım yazıları, sunum videoları, araçlar gibi kaynaklar) alınmıştır. Yazılım testlerinin otomatikleştirilmesinin zamanını ve neleri içereceğini etkileyen faktörler bu çalışmayla birlikte saptanmıştır. "Regresyon testleri için ihtiyaç", "ekonomik faktörler" ve "yazılımın olgunluğu" faktörlerinin, en fazla etkileyici faktörler olarak değerlendirildiği raporlanmıştır. Çok sayıda yazılım test otomasyon aracında, otomasyonu sağlamak için (örneğin, Selenium) betik (script) dillerinde çeşitli kodlar yazmak gereklidir. Test uzmanının yazılım geliştirme kökenli olmadığı durumda, betik kodu yazma zorunluluğu ortaya farklı güçlükler çıkarabilmektedir. Bunu aşmak üzere, bu tür yazılımlara, farklı eklentiler yazılarak, bazı işlemlerin var olan test durumlarını seçerek gerçekleştirilebilmesi sağlanabilmektedir. Örneğin, Selenium açık kaynaklı yazılıma bazı eklentilerle, betik kodu yazmadan test otomasyonu yapılabilmektedir. Bu çalışmada ise Xamarin Test Cloud ortamında bazı test durumları tarafımızdan geliştirilmiş olup bu test durumları, mobil uygulamaların testlerini kod yazımına gerek olmadan gerçekleştirebilmektedir.

Mobil uygulamaların test teknikleri konusunda Zein ve arkadaşları (2016), 79 empirik çalışmayı incelemiş ve bir sınıflandırma şemasına eşlemişlerdir [10]. Testleri; kullanılabilirlik testi, test otomasyonu, bağlam-duyarlılığı (context-awareness), güvenlik ve genel kategori olmak üzere 5 bölümde ele almışlardır. "Test otomasyonu" başlığı altında kullanılan teknikleri ise model tabanlı, veri güdümlü, taşınabilir işletim sistem kütüphaneleri, kara kutu, duyarlı-olay temelli, betikli kullanıcı arayüzü, ayrıntılı test kuvvetlendirme, tersine mühendislik, veri akış analizi, bağlamsal bulanıklandırma, makine öğrenmesi, yaklaşık yürütme, servis olarak otomatikleştirilmiş mobil test, paralel GUI testi, arama tabanlı, test kümelerinin sistematik incelenmesi, sensör ve olay-akışı temelli yaklaşım, sensör simulasyonu şeklinde kategorize etmişlerdir [11]. Kos ve arkadaşları (2016), alana özel modelleme dili (domain specific modelling language-DSML) tanımlayarak bir ölçüm sisteminin test otomasyonunu sağlamıştır [12]. Test otomasyonu alanında, DSML kullanmanın faydaları bu çalışmada ortaya konulmuştur. Tao ve arkadaşları (2015); "bulut temelli servis olarak mobil test" (Mobile TaaS) konusundaki temel kavramlara odaklanmış, beklenen altyapıyı açıklamışlardır [13]. Çalışmada farklı vaka çalışmaları ile önerilen yaklaşımın etkinliği ortaya konulmuştur. Collins (2012), endüstriyel bir deneyim çalışması olarak, çevik geliştirme ortamlarında test otomasyon pratiklerini açıklamışlardır [14].

3 YÖNTEM

Geliştirilen uygulamanın amacı Xamarin test bulutu üzerinde mobil test yapılabilmesi için uygulamalara yazılması gereken kod parçacıklarının gerekliliğini ortadan kaldırarak, test uzmanının çalışma yükünü hafifletmektir. Xamarin ortamında testlerin gerçekleştirilebilmesi için, mobil uygulamaların arayüzlerinde geçen her bir kontrol için Şekil 1'de gösterildiği gibi test durumu kodlanması gerekliliği bulunmaktadır. Test bulutu, yazılan test durumlarını referans alarak sanal makineler üzerinde testleri gerçekleştirmekte ve sonuçları web arayüzünden göstermektedir.

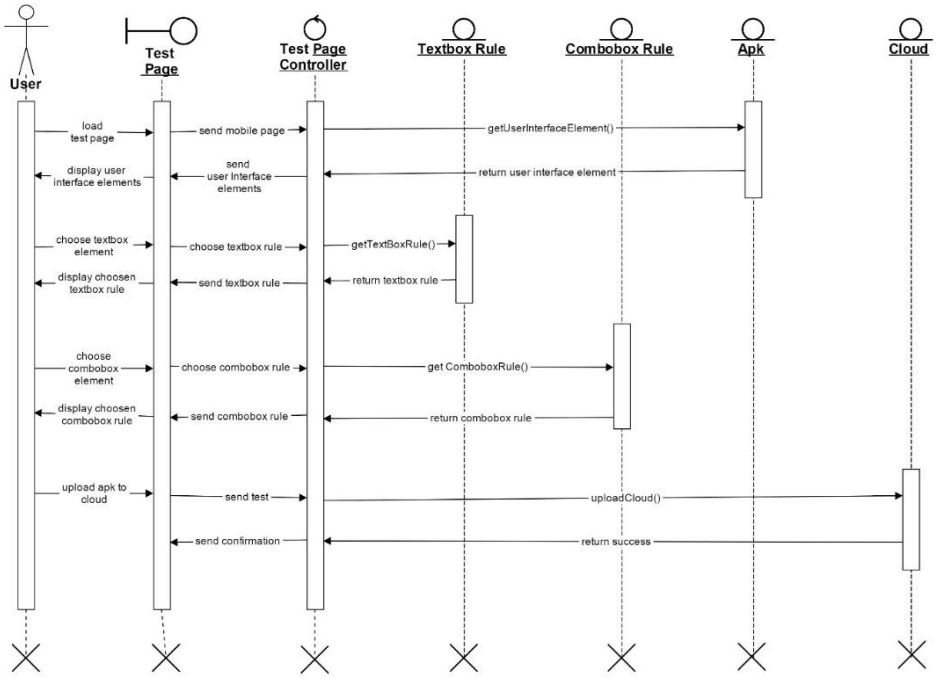


Şekil 1 : Test Durum Kod Parçacığı

Test uzmanın kod yazmadan sınanacak uygulamayı kontrol etmesi için alternatif olarak, Xamarin Test Kaydedicisini (Test Recorder) kullanabilir. Fakat, test kaydedicisinde, test uzmanın apk'yı en az bir kere bir sanal makine üzerinde çalıştırarak, gerçekleştireceği arayüz testlerini manuel olarak uygulamalıdır. Xamarin test bulutu, test uzmanın yaptığı işlemleri kayıt altına alarak, diğer farklı konfigurasyondaki sanal makineler üzerinde uygulayarak ve test sonuçlarını gösterecektir.

Her iki yöntemde de test uzmanın doğrudan uygulamayı çalıştırması gerekmekte veya kod yazması ihtiyacı bulunmaktadır. Önerilen yöntem ile, mobil uygulamalarda kullanılan kontrollerin test edilmesine uygun önceden hazırlanmış olan jenerik test durumları yazılmıştır. Bu şablon test durumları bir JSON dosyası içerisinde test uzmanına sunulmakta ve mobil uygulamaların arayüzleri test edilmektedir.

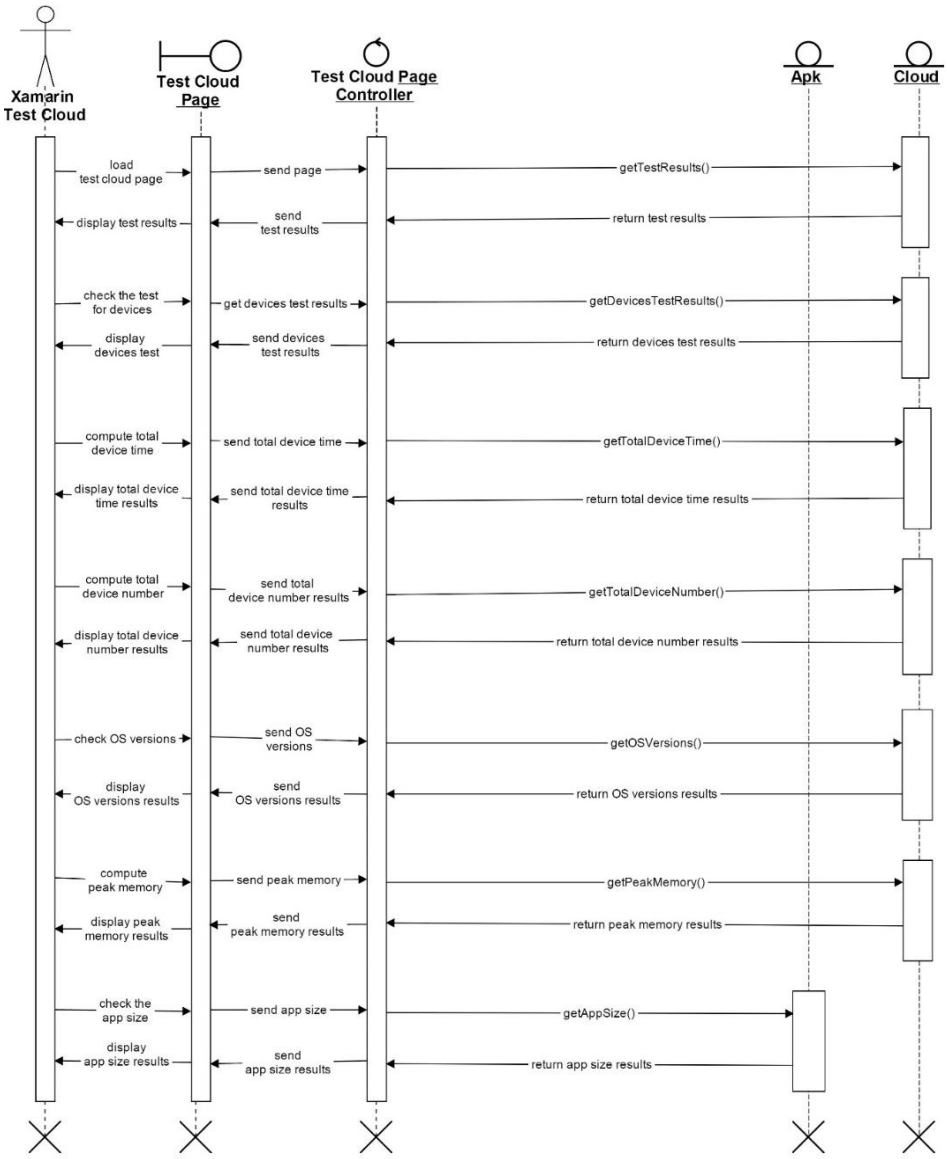
Test uzmanı, apk dosyasını Xamarin test bulutu üzerine yükledikten sonra, arayüzlerin içerisinde bulunan kontrolleri seçmektedir. Her farklı kontrol için daha önceden yazılmış olan test durum kuralları JSON dosyası içerisinden tetiklenerek sanal makineler üzerinde çalıştırılmaya hazırlanmaktadır. Apk içerisindeki tüm arayüzler için testi yapılacak olan tüm kontrollerin belirlenmesinin ardından her biri için uygun test durumu eşleştirilmesi ile test uzmanının görevi tamamlanmış olur. Şekil 2'deki sıralama diyagramında 2 farklı kontrolü barındıran bir arayüz için yapılacak işlemler sırasıyla gösterilmektedir.



Şekil 2 : Test Uzmanı İşlemleri Sıralama Diyagramı

Test kurallarının belirlenmesi sonrasında Xamarin test bulutu test edilecek cihazları *getTotalDevicesNumber* metodu ile belirler. Her bir cihaz üzerinde test işlemlerini yaptıktan sonra, test sonuçlarındaki en yüksek bellek kullanımı *getPeakMemory* metodu ile ve mobil uygulamanın kapladığı disk alanı *getAppSize* ile hesaplanarak tüm sonuçlar test uzmanına görüntülenir. Tüm bu iç işlemlere ait sıralama diyagramı Şekil 3'de gösterilmektedir.

Test uzmanın arayüzler üzerindeki kontrollere hangi verinin girileceği ve hangi işlemin yapılacağını yazdığı kod parçacıklarının ortadan kaldırılabilmesi için, ilk olarak sınanacak apk'nın tüm arayüzleri ve barındırdığı kontroller otomatik olarak tespit edilmektedir. Her bir girdi kontrolü (Text Fields) için arayüz testinde uygulanacak olan veri, gelişigüzel olarak alfa-numerik karakterlerden üretilmektedir. Veri üretilmesi gerekmeyen kontroller için (Checkboxes, Radio Buttons, Toggle Buttons, Spinners vb.) ise sistem bool mantığında bir sonuç üreterek test durumu için seçim oluşturur. Arayüz içerindeki tamamlayıcı kontrol (Button) ise en son olarak tetiklenerek arayüz testi, ileriki arayüzler için sürdürülür. Aynı uygulama içerisindeki benzerlik gösteren arayüzlerin testinde ve farklı cihazlar üzerinde yapılacak olan testlerde tekrar kullanılabilen bu ara-rutinler görev almaktadır.

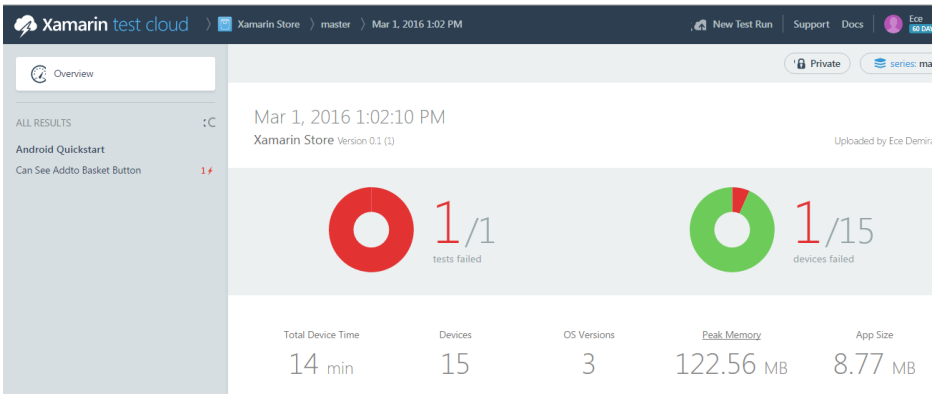


Şekil 3 : Xamarin Test Bulutu İşlemleri Sıralama Diyagramı

4 TEST

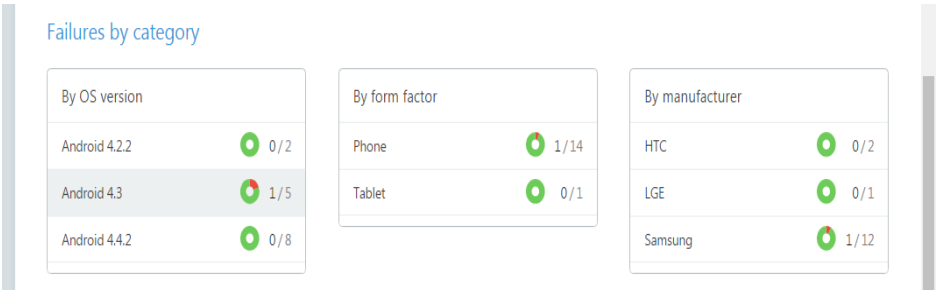
Önerilen yöntem ile açık kaynak kodlu AndroidQuickStart uygulaması bulut üzerinde test edilmiştir. Örnek uygulama kapsam itibariyle çevrimiçi t-shirt satış uygulamasıdır. Uygulama, katalog gösterimi ve sepet ile satış ekranlarını bünyesinde barındırmaktadır.

Test kapsamında farklı işletim sistemi sürümlerine sahip 15 farklı Android tabanlı cihaz üzerinde bulut testi yapılmıştır. 14 akıllı telefon ve 1 tablet üzerinde çalıştırılan apk'nın arayüz testlerinin kontrolleri 14dk'da tamamlanmıştır. Şekil 4'te görüldüğü üzere uygulamanın kapladığı disk alanı 8.77 MB olarak belirlenmiştir. Ayrıca testler esnasında tepe bellek kullanımında en yüksek tüketimin 122.56 MB olarak belirlendiği görülmektedir.



Şekil 4 : Xamarin Test Bulutu Sonuç Ekranı

HTC, LG ve Samsung marka belirlenen cihazlarda sırasıyla, Android 4.2.2, Android 4.3 ve Android 4.4.2 sürümlü işletim sistemleri yüklü olacak şekilde yapılan testler sonrasında, Android 4.3 işletim sistemine sahip Samsung Galaxy S5 cihazı üzerinde uygulamanın bellek yönetimi kaynaklı bir sorunla karşılaştığı tespit edilmiştir. Android işletim sistemlerinin üreticiler tarafından özelleştirildiği bu tarz durumlarda hata ile karşılaşma ihtimali artmaktadır. Şekil 5'de yapılan test süresince karşılaşılan hataların işletim sistemi, üretici ve yapıya göre kategorize edilerek gösterimi sunulmaktadır.



Şekil 5 : Xamarin Test Bulutu Sonuç Ekranı

5 SONUÇLAR ve GELECEK ÇALIŞMA

Mobil uygulamaların çok sayıda platform ve cihazda çalışabilmesi söz konusu olduğundan, her bir platformda ve cihazda uygulamanın davranışı test edilmeli, olası hatalar düzeltilerek tüm kullanıcıların sorunsuz şekilde kullanabileceği uygulamalar ortaya çıkarılmalıdır. Mobil uygulama testlerinin fazla oluşu, Xamarin Test Cloud gibi bulut ortamında çalışan test altyapısı yazılımlarının ortaya çıkmasını sağlamış, geliştiriciler açısından kapsamlı testler hızlıca yapılır duruma gelinmiştir. Bu altyapı yazılımları büyük kolaylıklar sağlasa da, kaynak kod yazımı gerektirmeleri nedeniyle, yazılım geliştirme alanında uzman olmayan test uzmanlarının işlerini zorlaştırmaktadır. Bu çalışmada, gerekli testler için kaynak kod yazımını ortadan kaldırılarak, şablon test durumlarının uygulanması sağlanmış olup test uzmanlarının iş yükü azalmış ve testler daha kısa sürelerde yapılır duruma gelmiştir. Bulut üzerindeki sanal sistemleri kullanarak yapılan otomatik arayüz testleri, test uzmanlarının gözünden kaçabileceği hataları da çok daha kısa sürede yakalayabilmektedir. Bu çalışma ile birlikte, Xamarin Test Cloud yazılımının, mobil uygulamaların test sürecinde önemli katkılar sunduğu tespit edilmiş, geliştirilen uygulama sayesinde kod yazımı gerekliliği minimum seviyeye çekilmiştir. Yazılım test mühendisliği alanında ülkemizde çalışan uzmanların bir kısmı, yazılım geliştirme altyapısına sahip olmadığı için bu tür kod yazımı gerektiren noktalarda sıkıntılar yaşamakta ve ilgili araçtan tam olarak yararlanamamaktadır. Bazı firmalar, bu nedenle kendi eklentilerini bu şekilde geliştirerek test uzmanlarının işlerini kolaylaştırmaya başlamıştır. Bulut ortamında çalışan Xamarin Test Cloud yazılımı için geliştirdiğimiz uygulama da benzer nitelikte katkılar sağlamış olup önümüzdeki dönemde yeni eklentilerle zenginleştirilecektir. Kavram ispatı olarak ortaya konulmuş olan ve faydaları görülen bu yaklaşımın, yeni test durumlarıyla birlikte sektörün önemli ihtiyaçlarına yanıt verebilir nitelikte olacağı değerlendirilmektedir. Yakın zamanda yeni test durumlarının eklenmesi planlanmıştır ve aktivite diyagramlarından tam otomatik bir şekilde testlerin üretilmesi mümkün olabilmesi için geliştirmeler devam etmektedir. Ayrıca, ticari mobil uygulamaların geliştirdiğimiz bu uygulama üzerinde testleri gerçekleştirilecek ve sektörün ihtiyacına dönük olarak beklentileri karşılama durumu incelenecektir. Bir yazılım firmasıyla görüşmeler gerçekleştirilmiş olup yakın zamanda geliştirdiğimiz uygulamanın, yeni geliştirilen mobil bir uygulama testlerini yapacak şekilde düzenlenmesi hedeflenmiştir.

6 REFERANSLAR

1. http://appium.io/ - Appium: Mobile App Automation Made Awesome
2. http://calaba.sh/ - Calaba.sh - Automated Acceptance Testing for iOS and Android Apps
3. https://github.com/robotiumtech/robotium - User scenario testing for Android
4. https://google.github.io/android-testing-support-library/docs/espresso/index.html - Build High-Quality Apps

5. https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html - I Automator is a UI testing framework suitable for cross-app functional UI testing across system and installed apps.
6. https://github.com/ajinabraham/Mobile-Security-Framework-MobSF - Mobile-Security-Framework (MobSF)
7. https://www.xamarin.com/test-cloud - Mobile App Testing On Hundreds Of Devices - Xamarin Test Cloud
8. Britton, T. Jeng, L. Carver, G., Cheak, P., & Katzenellenbogen, T. (2013). Reversible debugging software. *University of Cambridge-Judge Business School, Tech. Rep*.
9. Garousi, V. & Mäntylä, M. V. (2016). When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology,76*, 92-117.
10. Zein, S., Salleh, N., & Grundy, J. (2016). A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software, 117*, 334-356.
11. Hoffman, D. (1999). Test automation architectures: planning for test automation. In *Quality Week* (pp. 37-45).
12. Kos, T. Mernik, M, & Kosar, T. (2016). Test automation of a measurement system using a domain-specific modelling language. *Journal of Systems and Software, 111*, 74-88.
13. Tao, C. Gao, J. & Li, B. (2015, October). Cloud-Based Infrastructure for Mobile Testing as a Service. In *2015 Third International Conference on Advanced Cloud and Big Data* (pp. 133-140). IEEE.
14. Collins, E. F. (2012, June). Software test automation practices in agile development environment: An industry experience report. In *Proceedings of the 7th International Workshop on Automation of Software Test* (pp. 57-63). IEEE Press.