# Büyük-Ölçekli bir Android Uygulaması için Otomatik Fonksiyonel Test Yaratım Deneyimleri

# Automated Functional Test Generation Practice for a Large-Scale Android Application

Yavuz KÖROĞLU
*Bilgisayar Mühendisliği Bölümü*
*Boğaziçi Üniversitesi*
yavuz.koroglu@boun.edu.tr

Alper ŞEN
*Bilgisayar Mühendisliği Bölümü*
*Boğaziçi Üniversitesi*
alper.sen@boun.edu.tr

Abdurrahman AKIN
*Araştırma ve Geliştirme (Ar-Ge) Merkezi*
*Kuveyt Türk Katılım Bankası A.Ş., Kocaeli, Türkiye*
abdurrahman.akin@kuveytturk.com.tr

***Özet*—Pratikte yazılım geliştiriciler ve test ekipleri kullanıcı arayüzü otomasyon araçları yardımıyla fonksiyonel testler üretir. Bu araçlara rağmen, özellikle büyük-ölçekli uygulamalara fonksiyonel test yaratmak için geliştiriciler ve test ekipleri, (1) verili test senaryolarından test adımlarını üretmeli, (2) test adımlarını çalıştırmalı, ve (3) uygulamanın davranışını gözlemleyerek testin başarılı veya başarısız olduğuna karar vermelidir. FARLEAD-Android, tüm bu görevleri geliştiriciler ve test ekipleri yerine gerçekleştiren tam otomatik bir test yaratım motorudur. Bu ön çalışmamızda FARLEAD-Android motorunu büyük-ölçekli bir bankacılık uygulamasında çalıştırarak değerlendirdik. Deneyimizde FARLEAD-Android yaklaşık 24 dakika içerisinde uygulamadaki aktivitelerin %54,55 kadarına ulaşmakta, bu aktivitelerde altı farklı işlevin düzgün icrasını doğrulamakta ve doğrulama sırasında üç farklı işlevde ise sorunlar ortaya çıkarmaktadır. Değerlendirmemiz FARLEAD-Android motorunun daha etkili olması için deneyim tekrarı (experience replay) özelliğine sahip olması, aktivitelerin yanı sıra parçacıkları (fragments) da incelemesi ve kodlama bilmesi gerekmeyen üçüncü kişiler tarafından okunabilecek bir test senaryo biçemi kullanması gerektiğini göstermiştir.**

***Anahtar Terimler*—yazılım testi, mobil uygulamalar, test senaryoları**

***Abstract*—In practice, software developers and test teams generate functional tests with the help of user interface (UI) automation tools. Despite these tools, especially for large-scale applications, developers and test teams must (1) produce test steps from given test scenarios, (2) execute the test steps, and (3) evaluate if the test has passed or failed by observing the application behavior. FARLEAD-Android is a fully automated test generation engine which accomplishes these tasks for the developers and test teams. In this preliminary study, we evaluate FARLEAD-Android on a large-scale banking application. In our experiment, FARLEAD-Android reaches 54.55% of the activities, verifies the correct execution of six different functions, and reveals issues in three different functions during verification in approximately 24 minutes. Our evaluation shows that FARLEAD-Android must have an experience replay feature, investigate fragments along with activities, and use a human-readable test scenario format.**

***Index Terms*—software testing, mobile applications, test scenarios**

Bu çalışma BÜ Bilimsel Araştırma Projeleri tarafından 16201 kodu ile ve ITEA3 TESTOMAT (Test Otomasyonunun Gelecek Seviyesi) projesi (www.testomatproject.eu) tarafından desteklenmiştir.

978-1-7281-8541-5/20/$31.00 ©2020 IEEE

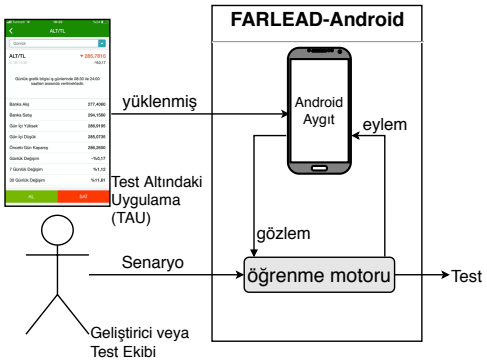
## I. GİRİŞ

Günümüzde mobil uygulama piyasasındaki uygulamaların %80'den fazlası Android Grafiksel Kullanıcı Arayüzü (GKA) uygulamalarıdır [1]. Yakın zamanda yapılmış bir anket çalışması [2] mobil GKA uygulama kullanıcılarının %78'inin, düzenli olarak kullandıkları uygulamaların işlevlerini engelleyen veya aksatan hatalarla karşılaştığını göstermektedir. Dolayısıyla Android uygulamalarının işlevlerini yerine getirebildiklerini doğrulayan testler yapmak gereklidir.

Fonksiyonel testler uygulamaların işlevlerini düzgünce yerine getirdiğini doğrulamak için kullanılan testlerdir. Pratikte yazılım geliştiriciler ve test ekipleri fonksiyonel testleri kullanıcı arayüzü otomasyon araçları kullanarak üretir. Özellikle büyük-ölçekli uygulamaların işlevlerini sınamak üzere geliştiriciler, teknik olmayan, kodlama bilmesi gerekmeyen üçüncü kişilerin okuyabileceği bir dilde test senaryoları hazırlarlar. Test senaryoları okunabilirliği yüksek tutmak amacıyla tipik olarak belirsiz test adımlarından oluşur. Geliştiriciler ve test ekipleri test otomasyon araçları yardımıyla bu belirsiz adımların her biri için kod yazarak icra edilebilir testler elde ederler. Geliştiriciler ve test ekipleri bu testleri test otomasyon ortamında çalıştırır ve uygulamanın davranışını gözlemleyerek testlerin başarılı olup olmadığını denetlerler. Test otomasyon araçları kullandıkları durumda bile geliştiriciler ve test ekipleri verili test senaryolarından (1) test adımlarını üretmeli, (2) test adımlarını çalıştırmalı, ve (3) uygulamanın davranışını gözlemleyerek testin başarılı veya başarısız olduğuna karar vermelidir.

FARLEAD-Android (Fully Automated Reinforcement Learning Driven-Android) [3] tam otomatik bir fonksiyonel test üretim motorudur. Şekil 1, bu motorun genel görünümünü içermektedir. Şekle göre öncelikle test altındaki Android uygulamasının bir Android aygıtında yüklü olduğu varsayılmaktadır. Geliştirici veya test ekibi ise bu Android aygıtı ile etkileşecek olan öğrenme motoruna test senaryosunu sağlamakla yükümlüdür. Öğrenme motoru test altındaki uygulamanın yüklü olduğu Android aygıtına eylemler yollayarak ve gözlemlerde bulunarak test senaryosunu gerçekleştirecek bir eylem dizisi öğrenir. Son olarak FARLEAD-Android test senaryosunu gerçekleştiren eylem dizisini, kısaca testi, çıktı olarak verir.

Test senaryolarını aynı zamanda test kahini olarak kullanması FARLEAD-Android'i diğer tam otomatik test üretim motorla-




Şekil 1. FARLEAD-Android'in Genel Görünümü

rından ayırır. Herhangi bir test üretim motoru da test adımları üretip çalıştırabilir ama FARLEAD-Android'in avantajları (1) test adımlarını test senaryolarına uygun biçimde yaratmayı öğrenebilmesi ve (2) test altındaki uygulamanın davranışını gözlemleyerek ürettiği testin senaryoyu başarıyla tamamladığına otomatik olarak karar verebilmesidir. Çalışmamızda FARLEAD-Android'i tercih etmemizin temel sebebi bu avantajlardır.

FARLEAD-Android test senaryolarına uygun test adımlarını destekli öğrenme (reinforcement learning) adı verilen yarı-gözetimli (semi-supervised) bir makine öğrenmesi yöntemi ile öğrenir. Destekli öğrenmenin yapay sinir ağları gibi diğer yapay zekalara göre avantajı önceden etiketlenmiş ve filtrelenmiş bir veri gerektirmemesi, bunun yerine deneme-yanılma ile öğrenmesidir. Destekli öğrenme kaynak yönetiminden [4] trafik ışığı kontrolüne [5], satranç [6] ve atari [7] gibi karmaşık oyunlardan kimyaya [8] kadar çok geniş bir alanda insan becerisinin ötesine geçmiştir. FARLEAD-Android'in bu uygulamalara göre ek bir avantajı ise hedef testleri ideal stratejiye (policy) yakınsamadan önce üretebilmesi ve böylece test üretimini kısa bir sürede gerçekleştirmesidir [3]. Böylece FARLEAD-Android'in büyük-ölçekli uygulamalarda kullanım olanağı doğmuştur.

Bu çalışmamızda büyük-ölçekli bir mobil bankacılık uygulaması için FARLEAD-Android'e uygun 31 test senaryosu ürettik. FARLEAD-Android'i yaklaşık 24 dakika çalıştırdık. FARLEAD-Android'in doğruladığı işlevleri, ortaya çıkardığı sorunları, ve kapsadığı aktiviteleri ölçtük. Karşılaştığımız problemleri ve FARLEAD-Android'in eksikliklerini raporladık. Bu sayede FARLEAD-Android'i pratikte kullanılan bir test yaratım çözümü haline getirmek konusunda ilerleme kaydetmeyi amaçlıyoruz. Bildiğimiz kadarıyla, bu çalışma büyük-ölçekli bir mobil uygulamanın fonksiyonel testlerinin tam otomatik yaratımı deneyini gerçekleştiren ilk çalışmadır. Literatürde bu çalışmamıza en yakın STAMBA [9] aracı bulunmaktadır. STAMBA çalışması bir mobil bankacılık uygulamasının sadece belli güvenlik olanaklarını test etmeye özelleşmiştir ve FARLEAD-Android gibi fonksiyonel test üretmemektedir.

## II. YÖNTEM

### A. Ön Hazırlıklar

Bu çalışmamızda büyük-ölçekli bir mobil bankacılık uygulamasını test ettik. Bu uygulama 22 aktivitesi olan, yaklaşık 45,7 MB boyutunda, büyük-ölçekli bir uygulamadır. Deneyimiz için öncelikle 22 aktivitenin her biri için o aktiviteye ulaşılması gerektiğini belirten birer test senaryosu hazırladık. Daha sonra, her bankacılık uygulamasında gerçekleşmesini

Tablo I
ÖRNEK DZM SENARYOSU

$p = [\text{activity} \sim \text{IbanCalculation}]$
$q = [\text{actionType} = \text{click}] \wedge [\text{actionDetail} \sim \text{"Obtain Account No"}]$
$r = [\text{actionType} = \text{text}] \wedge [\text{actionParam} = \text{"12345678912345678912345 6"}]$
$s = \neg[\text{text} \sim \text{"TR123456789123456789123456"}]$

$\phi = \bigcirc\Diamond[p \wedge \bigcirc\Diamond[q \wedge \bigcirc[r \wedge s]]]$

beklediğimiz işlevler için altı ve olası sorunlar için üç ayrı test senaryosu oluşturduk. Adına senaryo oluşturduğumuz işlevler (1) belirli miktar TL'yi altına çevirmek, (2) döviz kurlarını anlık güncelleyebilmek, (3) giriş yapmak, (4) giriş yaptıktan sonra yeterince beklendiğinde otomatik çıkış yaptığını gözlemlemek, (5) IBAN hesaplayıcıya başka bankanın IBAN'ı verildiğinde uyarı mesajı görmek, ve (6) IBAN kısmına yazı yazamamaktır. Adına senaryo oluşturduğumuz sorunlarsa (1) IBAN kısmına sadece rakam girildiğinde *TR* ön ekinin otomatik konulmaması veya yanlış konulması, (2) çok yüklü miktarda altını değerinden düşük fiyata dönüştürmesi (overflow), ve (3) GKA bileşenlerinin ekrana sığmaması sonucu dışarı taşması durumudur. Sonuç olarak toplamda 22 aktivite, 6 işlev, ve 3 sorun olmak üzere toplamda 31 adet test senaryosu elde ettik.

FARLEAD-Android test senaryolarını Doğrusal-zamanlı Zamansal Mantık (Linear-time Temporal Logic, DZM) formülleri olarak alır. DZM, önermeli mantığın zamansal operatörler ile genişletilmiş bir halidir. FARLEAD-Android deneme-yanılma ile verili bir DZM formülünün ne kadarının gerçekleştirildiği ile doğru orantılı bir ödül mekanizması ile formülün tamamını gerçekleştirmeyi öğrenir.

Tablo I, DZM formülü halinde yazılmış örnek bir senaryo göstermektedir. Bu örnek senaryoda IBAN hesaplama sayfasında yazılan IBAN'ların otomatik olarak tamamlanıp tamamlanmadığı test edilmektedir. Bu senaryo test altındaki uygulamanın *TR* ön ekini koyması gerekirken hataya düşmesini beklemektedir.

Test üretimi sırasında karşılaştığımız en büyük sorun FARLEAD-Android'in test senaryolarını DZM biçeminde almasıdır. Tipik bir test senaryosu teknik detayı düşük ve kodlama bilmesi gerekmeyen üçüncü kişiler tarafından okunabilir olmalıdır. Oysa DZM formülleri bunun tam tersidir. FARLEAD-Android'in pratikte kullanılabilmesi için Gherkin [10] gibi geliştiricilerin yaygın olarak kullandığı okunabilir test senaryolarından test üretebilmesi gerekmektedir.

### B. Deney Ortamı

FARLEAD-Android'i 1476 saniye (yaklaşık 24 dakika) boyunca 480x800 çözünürlüklü bir VirtualBox sanal Android makinesi üzerinde çalıştırdık. Deneylerimizi aynı özelliklere sahip bir aygıt kullanmadan tekrarlayabilmek için sanal makine tercih ettik.

Her test senaryosu için FARLEAD-Android parametrelerini azami adım sayısı ($K$) hariç özgün makaledeki gibi kullandık ve sabit tuttuk [3]. $K$ değerini ise her senaryo için yeniden belirleyerek FARLEAD-Android motoruna müdahale etmemiz gerekti. FARLEAD-Android'in kullanılabilirliğini artırmak için azami adım sayısını motora müdahale etmek yerine test senaryosu hazırlama sırasında belirtebileceğimiz genişletilmiş, standart bir test senaryosu diline gereksinim vardır.

Deney boyunca FARLEAD-Android'in hangi aktiviteleri kapsadığını, hangi işlevleri doğruladığını, ve hangi sorunları ortaya çıkardığını ölçtük. Bunların yanında Tablo II içerisinde

Tablo II
DENEY SONUÇLARI

| # | Aktivite Adı | Varıldı? | Ulaşma Süresi | Test Süresi | İşlev Sayısı | Sorun Sayısı |
|---|---|---|---|---|---|---|
| 1 | WelcomeSplash | ✔ | 24 | 24 | | |
| 2 | Introduction | ✔ | 33 | 33 | | |
| 3 | WelcomePage | ✔ | 42 | 169 | 1 | 1 |
| 4 | FxList | ✔ | 51 | 103 | 1 | |
| 5 | YourBankShowCase | ✔ | 53 | 53 | | |
| 6 | CalculationTools | ✔ | 48 | 175 | 1 | 1 |
| 7 | IbanCalculation | ✔ | 61 | 256 | 2 | 1 |
| 8 | FinanceCentral | ✔ | 52 | 52 | | |
| 9 | Info | ✔ | 53 | 53 | | |
| 10 | NearestBranch | ✔ | 49 | 49 | | |
| 11 | BranchDrawer | ✔ | 78 | 458 | 1 | |
| 12 | YourBank | ✔ | 51 | 51 | | |
| 13 | FaceID | ✖ | | | | |
| 14 | QRCodeTransactions | ✖ | | | | |
| 15 | MPos | ✖ | | | | |
| 16 | SendMoney | ✖ | | | | |
| 17 | QMatic | ✖ | | | | |
| 18 | CropImage | ✖ | | | | |
| 19 | barcodescanner | ✖ | | | | |
| 20 | GoogleApıActivity | ✖ | | | | |
| 21 | CardIO | ✖ | | | | |
| 22 | DataEntry | ✖ | | | | |
| | **TOPLAM** | %54,55 | 595 | 1476 | 6 | 3 |

Tüm süreler saniyedir.

test altındaki uygulamanın aktivitelerini FARLEAD-Android'in test üretimi sırasında keşfettiği Dinamik Aktivite Geçiş Grafiği'ni (DAGG) baz alarak uygulamanın ilk başlatıldığı duruma yakınlıklarına göre yakından uzağa doğru sıraladık.

## III. BULGULAR

Tablo II, hazırladığımız 22'si aktivitelere ulaşma, altısı işlevlerin doğrulanması, ve üçü sorunlarla alakalı olmak üzere toplam 31 test senaryosu için FARLEAD-Android'in test üretimi sonuçlarını göstermektedir. Bu sonuçlara göre FARLEAD-Android test altındaki uygulama üzerinde yaklaşık 24 dakika (1476 saniye) çalışmış ve bu sürede kendisine verilen altı işlevin tamamını doğrulamış, üç sorunun tamamını ortaya çıkarmıştır, ve 22 aktivitenin 12'sine (%54,55) ulaşmıştır. Böylece 31 test senaryosunun 21'ini gerçekleştirmiştir.

FARLEAD-Android test senaryolarının gerçekleştirildiği aktivitelere ulaşmak için yaklaşık 10 dakika (595) saniye harcamıştır. FARLEAD-Android her yeni test senaryosuna geçtiğinde bir önceki test senaryosunda öğrendiği adımları unuttuğu için uygulamanın derinlerinde olan aktivitelere gitmek için daha önceden gitmeyi öğrenmiş olduğu yakın ara aktivitelere gitmeyi baştan keşfetmektedir. Bu durum aktiviteler arası geçişleri öğrenmek için FARLEAD-Android'in çok uzun süre harcamasına neden olmaktadır. Test altındaki uygulama açıldığında ekranda gösterilen ilk aktivite *WelcomeSplash* aktivitesidir. Bu aktivitenin açılması için FARLEAD-Android'in harcadığı 24 saniye sadece uygulamanın yüklenmesi için geçen süredir. Baştan uygulamayı çalıştırıp ikinci aktivite olan *Introduction* aktivitesine ulaşmayı keşfetmek FARLEAD-Android'in 33 saniyesini almıştır. Baştan üçüncü aktiviteyi keşfetmek ise 42 saniye sürmüştür. FARLEAD-Android her senaryoda daha önce keşfettiği aktiviteleri baştan keşfetmek için vakit harcamıştır. 4-12 nolu aktiviteler ise 3 nolu aktiviteden ulaşılabilen aktivitelerdir. Ancak daha derinde bulunan, doğru şifre ile giriş yapmış olmak gibi karmaşık işlemler gerektiren 13-22 nolu aktivitelere ise FARLEAD-Android ulaşamamıştır.

FARLEAD-Android'in derindeki aktivitelere ulaşabilmesi için önceki aktivitelere gidecek adımları aklında tutabilmesi gereklidir. Bu amaçla destekli öğrenmede deneyim tekrarı (experience replay) olarak bilinen yöntemin FARLEAD-Android'de uygulanması gerekmektedir [11]. Test senaryolarının deneyim tekrarından azami ölçüde yararlanacak şekilde sıralanmasına dair bir çalışmayı ise gelecekte yapmayı planlamaktayız. Son bir not olarak vakit kazanabilmek için deneyim tekrarının aygıt üzerinde değil, FARLEAD-Android'in hafızasında simülasyon ile gerçekleştirilmesi gerekmektedir. İleriki çalışmalarımızda bu problemi de çözmeye çalışacağız.

FARLEAD-Android'in test sırasında ulaştığı 12 aktivitenin 11'i uygulamaya şifre ile giriş yapmadan önceki işlevleri içermektedir. Test altındaki uygulamanın *jdaxgui* adı verilen statik analiz aracı yardımıyla detaylı incelenmesi bankacılık işlemlerinin büyük çoğunluğunu giriş yaptıktan sonra açılan *BranchDrawer* aktivitesinin altında dinamik olarak açılıp kapanabilen parçacıklar (fragments) yardımıyla yapıldığını ortaya koymuştur. Dolayısıyla ileride aktivite kapsaması yerine literatürde henüz yeni yer almaya başlamış olan parçacıklar [12] üzerinden test senaryoları tanımlamayı planlamaktayız.

Gherkin [10], DZM formüllerinin aksine okunabilir test senaryoları üretmek için idealdir, ama belirsizliği yüksektir ve $K$ gibi detaylar bu dilde belirtilememektedir. FARLEAD-Android'e uygun ve okunabilirliği yüksek bir test senaryosu dilini ileride önermeyi amaçlamaktayız.

Özetle, bu ön çalışmada FARLEAD-Android motorunu büyük-ölçekli bir bankacılık uygulamasında çalıştırarak değerlendirdik. Deneyimizde FARLEAD-Android yaklaşık 24 dakika içerisinde uygulamadaki aktivitelerin %54,55 kadarına ulaşmış, bu aktivitelerde altı farklı işlevin düzgün icrasını doğrulamış ve üç farklı işlevde ise sorunlar ortaya çıkarmıştır. Değerlendirmemiz FARLEAD-Android motorunun daha etkili olması için deneyim tekrarı (experience replay) özelliğine sahip olması, aktivitelerin yanı sıra parçacıkları (fragments) da incelemesi ve azami adım sayısının belirlenebildiği, kodlama bilmeyen üçüncü kişilerin okuyabileceği bir test senaryo biçemi kullanması gerektiğini göstermiştir.

## KAYNAKLAR


[1] "Market share: Final pcs, ultramobiles and mobile phones, all countries, 4q17 update," 2018.

[2] D. Bolton, "88 percent of people will abandon an app because of bugs," 2017, https://www.applause.com/blog/app-abandonment-bug-testing.

[3] Y. Koroglu and A. Sen, "Reinforcement learning-driven test generation for android gui applications using formal specifications," *arXiv preprint arXiv:1911.05403*, 2019.

[4] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 50–56.

[5] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] Z. Zhou, X. Li, and R. N. Zare, "Optimizing chemical reactions with deep reinforcement learning," *ACS central science*, vol. 3, no. 12, pp. 1337–1344, 2017.

[9] S. Bojjagani and V. N. Sastry, "Stamba: Security testing for android mobile banking apps," in *Advances in Signal Processing and Intelligent Recognition Systems*. Cham: Springer International Publishing, 2016, pp. 671–683.

[10] "Gherkin language," https://en.wikipedia.org/wiki/Cucumber_(software).

[11] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[12] J. Chen, G. Han, S. Guo, and W. Diao, "Fragdroid: Automated user interface interaction with activity and fragment analysis in android applications," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 398–409.