Model Bazlı Test Yöntemiyle Açık Bankacılık Ödeme Sisteminin Testi
Testing OpenAPI Banking Payment System with Model Based Test Approach

Ahmet Sapan, Bilgehan Öztekin, Ersin Ünsal
Ar-Ge Merkezi
Fibabanka
ahmet.sapan, bilgehan.oztekin, ersin.unsal@fibabanka.com.tr

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

***Özet*—Model Bazlı Test (MBT), yazılım testlerinde yaygın olarak kullanılan bir yaklaşımdır. Modeller test edilen sistemin işlevselliğini ve davranışını açıklar. Bu nedenle geliştirilen model etkili test senaryolarının oluşturulmasını kolaylaştırır. Bu çalışmada, bir MBT aracı olan GraphWalker kullanılarak Açık API tabanlı bir ödeme sisteminin test sürecinde nasıl kullanıldığı anlatılmaktadır.**

***Anahtar Kelimeler*—Model Bazlı Test, GraphWalker, Selenium Web Driver, Model Tabanlı Ödeme Sistemi**

***Abstract*—Model Based Testing (MBT) is one of the common approaches in software testings. A model explains characteristics and the functionality of the system tested. Therefore, having developed models ease the process of developing effective test scenario. This article shows how behaviour driven test automation developed with GraphWalker is used in the test process of an OpenAPI based payment system.**

***Keywords*—Model Based Testing, GraphWalker, Selenium Web Driver, Model Based Payment System**

I. GİRİŞ

Model Bazlı Test (MBT) tekniği, modellerin yardımıyla ihtiyaçlardan doğan test senaryolarının üretilmesinde kullanılır [1]. Bir başka deyişle, yazılım davranışları modelde tanımlanmış davranışlarla karşılaştırılır. Genel MBT şeması Şekil 1'de görülebilir. Burada model, yazılım geliştirici ve test uzmanlarının hazırladığı gerçek dünya sisteminin soyutlanmış halidir. Yazılım geliştirici, test uzmanı ya da otomatik bir araç sistem modelinden soyut testler türetir. Yine yazılım geliştirici, test uzmanı ya da otomatik bir araç, soyut testlerden yürütülebilir testler (executable tests) üretir. Son olarak ise yürütülebilir testler, dinamik işletim (dynamic execution) ile sistemin doğru çalıştığını teyit eder ya da yazılım hatalarını ortaya çıkarır.

Bazı durumlarda [2]–[5], modelden doğrudan yürütülebilir testler üretilmesi mümkündür. Soyut testlere sahip olmak beraberinde bazı avantajları getirir. Örneğin, ReCDroid [6] yazılım hatası raporlarından sağlanan test adımlarını soyut test olarak kullanır. ReCDroid otomatik olarak soyut testlerden yürütülebilir testler üretir. Bir başka test üretim motoru olan FARLEAD-Android [7], yürütülebilir testlerini Gherkin dilinde [8] yazılmış kullanıcı dostu test senaryolarından üretir.

Şekil 1. Genel MBT Şeması

ReCDroid ve FARLEAD-Android'teki soyut testler, kullanıcıların kaynak kod ya da genel kodlama ile ilgili bilgi sahibi olmadıkları halde neyin test edildiğini görüntüleyebilmelerine izin verir. Buna ek olarak soyut testler; yazılım geliştiricinin ya da test uzmanının, ortam spesifik detayları test senaryosundan gizlemesine olanak sağlar; böylece test başka cihazlarda ve işletim sistemlerinde farklı konfigürasyonlarla uygulanabilir. Son olarak, soyut testler yürütülebilir testlere nazaran kaynak kodlardaki küçük değişikliklere karşı daha dayanıklı olduğu için, soyut testler daha iyi sürdürülebilirliğe sahiptir.

Yazılım geliştiriciler genellikle Grafik Kullanıcı Arayüz (GUI) uygulamaları için modeller yaratmazlar, bu yüzden sistem modelinden test üretmek mümkün olmayabilir. Bu durum özellikle pazarın (GUI modellemesi için ekstra zaman harcamak için) çok rekabetçi olduğu Android GUI uygulamaları için doğrudur [9]. Böyle durumlarda, otomatik test üretim aygıtları çoğu zaman dinamik işletim aracılığı ile GUI modeli öğrenirler [10][11].

Açık Bankacılık (Open API) sayesinde bankacılık sektörü dönüşüm yaşamaktadır. Açık bankacılık banka müşterisine API'ler aracılığıyla, diğer tüm finansal kuruluşlardaki verilerine ulaşım imkanı veren bir ağ sistemdir. Bu çalışmanın konusu olan Model Tabanlı Ödeme Sistemi (MOTES) bu açık API'leri kullanarak müşterilere yeni bir ödeme yapma deneyimi sağlar. Model Tabanlı Ödeme Sistemi'nin Model Bazlı Test yöntemi yardımıyla nasıl test edildiği anlatılmıştır.

II. Model Tabanlı Ödeme Sistemine Genel Bakış

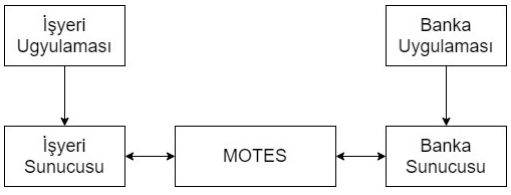
Model Tabanlı Ödeme Sistemi (MOTES), e-ticaret siteleri üzerinden çeşitli ödeme yöntemleriyle ödeme yapılmasını sağlayan bir yazılım ürünüdür. MOTES; müşterilere, ödeme yapmak istediği banka ya da mobil cüzdan uygulamasına bağlanarak kolay bir şekilde ödeme yapma imkânı sağlar. MOTES, anlaşmalı bankaların API servislerine bağlanarak fonksiyonlarını gerçekleştirir. Son dönemde bankaların açık bankacılık fikrini benimselemeleri, API'ların fintechler tarafından daha sık kullanılmalarını sağladı. Ek olarak MOTES, işyerleri ve bankalar arasında finansal işlemler gerçekleştirdiği için sürekli aktif halde çalışır. Kapasitesi işlem sayısının yüksek olduğu günler için yeterli olmalıdır.

Müşterilere sağladığı kolaylığın yanı sıra MOTES, müşterilere ödemelerini güvenli bir şekilde gerçekleştirme imkânı da sağlar. Müşteriler, MOTES ödeme sayfasına yönlendirildiğinde; MOTES ödeme işlemini çevrimiçi olarak ilgili bankanın ya da mobil cüzdanın sistemine yönlendirir. Müşteriler kart bilgilerini ya da şifrelerini MOTES ile paylaşmak zorunda değildir, MOTES müşteri ile müşterinin üzerinden ödeme yapmak istediği banka arasında bir köprü işlevi görür. Bankadan ya da mobil cüzdandan gelen işlem cevabını da işyeri ile paylaşır.

MOTES'in bir diğer avantajı da bankaların e-ticaret sitelerine entegre olmasını kolaylaştırmasıdır. Bankaların iş yerlerinin hepsine ayrı ayrı entegre olması ciddi kaynak, emek ve zaman gerektiren bir iştir. MOTES ise, bankaları bu ciddi zahmetten kurtarır. Bankalar ve iş yerleri MOTES'e entegre olurlar ve böylece MOTES, bankalar ile iş yerleri arasındaki entegrasyon sürecini maksimum derecede kolaylaştırmış olur. MOTES entegrasyon servisleri; ödeme, iade, loglama ve admin paneli üzerinden kullanıcı operasyonlarının gerçekleştirildiği sınıfları içerir. Ayrıca entegrasyon sürecinde bankaların testlerini gerçekleştirebilmeleri için, MOTES işyeri test simülasyonu oluşturulmuştur.

MOTES bankaların API servislerine e-ticaret siteleri üzerinden yapılan alışverişin bilgilerini iletir. Müşteri MOTES tarafından doğrudan ilgili bankanın mobil ödeme ekranına yönlendirilir ve banka, ödeme sonucu bilgilerini MOTES'e daha önceden belirlenen kodlarla iletir. MOTES, işlem sonucunun olumlu ya da olumsuz olduğunu işyerine iletir.

MOTES'in yüksek seviye sistem mimarisi tasarımı Şekil 2'de görülebilir.



Şekil 2. Yüksek Seviye Sistem Mimarisi

III. Graphwalker ve Selenium

Üretilen Model Bazlı Test, Graphwalker ve Selenium araçları kullanılarak geliştirilmiştir.

GraphWalker [12], açık kaynaklı bir model bazlı test aracıdır. Sahip olduğu üç ana özellik ile test süreçlerini büyük ölçüde geliştirir. GraphWalker, Studio adında üzerinde modellerin oluşturulabileceği ve düzenlenebileceği bir editör sunar. Studio ile modeller test izleri (test path) üretilerek doğrulanabilir, böylece kullanıcı modelin hangi ölçüde doğru çalıştığını görebilir. Komut satırı araçları kullanılarak çevrimiçi/çevrimdışı test izi üretimi yapılabilir.

Bir GraphWalker modeli, düğüm (node) ve kenar (edge) olmak üzere 2 temel öğeden oluşur. Kenar, bir eylemi ve geçişi temsil eder. Bir eylem, API çağrısı, button click, zaman aşımı (timeout) vb. olabilir. Oluşturduğunuz testi doğrulamak istediğiniz yeni bir duruma taşıyan işlemlerin tümü kenarda gerçekleşir. Kenarlarda hiçbir doğrulama yoktur. Bu sadece düğümlerde yapılabilir. Düğümler bir doğrulamayı, bir iddiayı (assertion'ı) temsil eder. Doğrulama, kodunuzda iddialarınızın bulunduğu yerdir. Burada, bir API çağrısının doğru değerleri döndürdüğünü, bir düğme tıklamasının gerçekten bir iletişim kutusunu kapattığını veya zaman aşımının ne zaman gerçekleşmesi gerektiğini doğrularsınız.

Graphwalker ile oluşturulan soyut modelden test şablonları üretildikten sonra bu şablonların içeriği Selenium kütüphanesi kullanılarak doldurulur. Grafik arayüzlerinin testinde en çok kullanılan uygulamalardan birisi de Selenium kütüphanesidir. Selenium WebDriver [13], tarayıcıları işletim sistemi düzeyinde yapılandırabilen ve kontrol edebilen bir platformlar arası test çerçevesidir. Test senaryoları oluşturmak ve çalıştırmak için bir programlama arabirimi olarak hizmet eder. WebDriver web öğeleri üzerinde eylemler gerçekleştirir. Java, C , PHP, Python gibi çeşitli programlama dillerini destekler. Selenium WebDriver'ın test sonuçları oluşturmak için yerleşik özelliği yoktur. Test raporlarının oluşturulması GraphWalker gibi üçüncü taraf araçlarına bağlıdır. Test yönetimi için TestNG ve JUnit gibi çerçevelerle de entegre edilebilir.

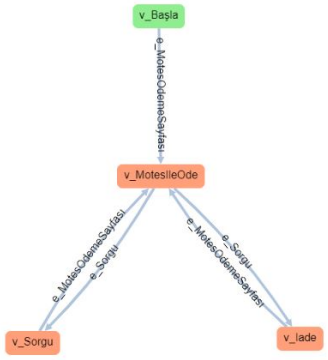
IV. Bulgular

Bu bölümde, Graphwalker kullanılarak geliştiridiğimiz modeller ve çalışma sonuçları yer almaktadır.

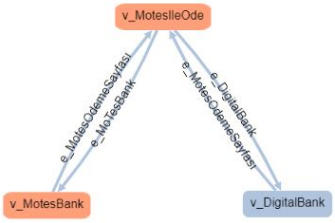
En üst seviye de modelimiz Şekil 3'de görülmektedir. Model v_Başla başlangıç düğümünden testin yapılmak istendiği tarayıcı açıldıktan sonra MOTES'in ödeme akışını başlatan sayfasına yönlenir. Bu sayfa MOTES'e ödeme bilgilerinin iletildiği sayfadır ve sepet tutarı, ürün bilgileri, ödemenin yapılacağı banka numarası vb. birçok input alanını içermektedir. Ödeme bilgileri girildikten sonra bu sayfada "gönder" butonuna basıldığında bu bilgiler JSON formatında MOTES'e iletilmektedir.

Ana model, v_MotesİleÖde düğümüne ulaştığında akış, Şekil 4'de bulunan model ile devam etmektedir. Şekil 4'de bulunan v_MotesİleÖde ile aynı isme sahip ana modelde bulunan düğüm, ortak bir "Shared Name" değerine sahiptir.

Model bu noktadan sonra, ödemenin yapılacağı banka ile devam etmektedir. Bir önceki adımda girilen parametrelere göre seçilen banka MOTES ile entegrasyonu tamamlanmış bir dijital bankaya ait ise ve girilen parametreler ile seçili bankadan başarılı bir şekilde token alınabildiyse bu bankaya ait inter-

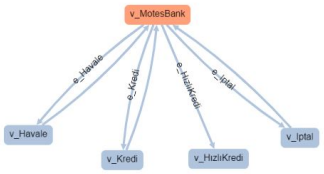


Şekil 3. Ana model



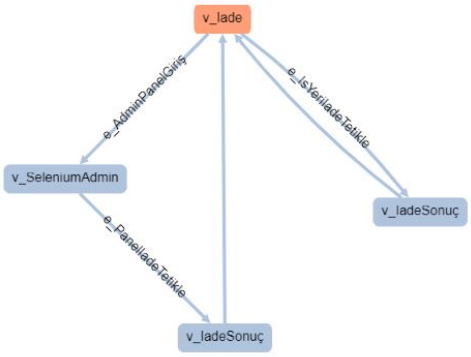
Şekil 4. MOTES ile Öde

net bankacılığının sayfasına yönlenme gerçekleşir. Yönlenme gerçekleştikten sonra işlem tamamlanmaz ve ödeme sayfasına geri yönlenme gerçekleşir. Daha sonra model, akışı tekrar ana modele yönlendirip Şekil 3'teki v_İade veya v_Sorgu ile devam ettirebilir ya da aynı modeldeki v_MotesBank ile devam ettirebilir. v_MotesBank ile devam edildiği durumda akış Şekil 5'deki model üzerinden devam eder. Bu modele gelindiğinde MOTES'in bir parçası olan banka simülasyon sayfası açılmış olmalıdır.



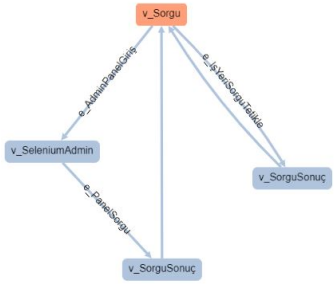
Şekil 5. Motes Simülasyon Bankası

Bu sayfada işlemin nasıl sonuçlandırılması gerektiği belirlenir. İşlem tipi havale, kredi veya hızlı kredi seçilerek başarılı tamamlanabilir veya iptal seçeneği ile işlem iptal edilebilir. Tamamlanan işlemler bir listede tutulur. Bu liste işlem numarası, işlem durumu ve kalan miktar (kısmi iade işleminden sonra) özelliklerine sahip Java objelerinden oluşmaktadır. Bu bilgiler daha sonra ana modelde bulunan iade ve durum sorgulama sırasında kullanılacaktır. Ödeme işleminin tamamlanmasının ardından akış v_MotesBank'ın çalışması ile devam eder. Bu düğümde, işlemin ne şekilde sonuçlandığı ile gerçekte sonuçlanmasını beklediğimiz değeri arasında bir doğrulama yapılır. Doğrulama adımından sonra akış ana modele geri dönüp iade veya işlem sorgusu başlatabilir ya da ana modele geri döndükten sonra tekrar ödeme işlemi başlatabilir. Ana modele geri dönüp işlem sorgusu başlatıldığı durumda akış Şekil 6' daki model üzerinden devam eder.



Şekil 6. İade Servisi

MOTES ile 2 tip iade tetiklemesi yapılabilir. Birincisi admin paneli aracılığı ile, ikincisi ise iş yeri simülasyonunda bulunan sorgu parametrelerini kullanarak. Admin paneli aracılığı ile iade yapılması için önce sisteme giriş yapılması gerekmektedir. Başarılı bir şekilde giriş yapıldıktan sonra admin panelden işlem kayıtlarının bulunduğu sekme seçilir. Bu sekmedeki işlem numarası alanına test süresince denenmiş olan işlemlerden birinin işlem numarası girilerek bu işlemin paneldeki loglarda görülen işlem durumu ile mevcut işlem durumu arasında doğrulama yapılır. Sorgu servisinin tamamlanmasının ardından akış daha önceki yolları tekrar edebilir veya iade servisini çalıştırabilir. İade servisini çalıştırdığı durumda Şekil 7'deki model üzerinden devam eder.



Şekil 7. Sorgu Servisi

Bu model, MOTES'in hem panel hem de iş yeri simülasyon sayfasından iade yapmayı desteklemesi sebebiyle sorgu servisi ile benzerlik taşır. Admin panelden yapılan iadeler için, panele giriş yapılarak işlem durumu "TAMAMLANDI" olarak güncellenmiş olan işlemler aratılır. Bu işlemler üzerinden iade yapılır ve işleme ait kalan tutar değeri güncellenir. İşleme ait kalan tutar değeri 0'a ulaşana kadar hem panelden hem iş yeri simülasyon sayfasından iade işlemi yapılabilir.

Modeller hazırlandıktan sonra oluşturulan test şablonlarının içeriği Selenium kütüphanesi kullanılarak doldurulur. Örnek

bir Selenium kodu aşağıda görülmektedir.

```
@Override
public void v_HizliKredi() {
    System.out.println("Executing:v_HizliKredi");
    WebElement webElementState = driver.findElement(By.name("Response"));
    String response = webElementState.getAttribute("value");

    Assert.assertEquals(response,"Pending");
}

@Override
public void e_HizliKredi() {
    System.out.println("Executing:e_HizliKredi");

    Select dropdownPayment = new Select(driver.findElement(By.name("productType")));
    dropdownPayment.selectByVisibleText("Hızlı Kredi");
    driver.findElement(By.name("submit")).click();
    paymentRequestList.get(paymentCounter-1).setState("CREDIT_INIT");
}
```

Şekil 8. Selenium Kodu

Bu şekilde, Motes Bankası aracılığı ile e_HızlıKredi kenarında tamamlanması sağlanan "Hızlı Kredi" işlemi ve hemen sonrasında açılan işlem sonucu sayfası üzerinde uygulanan iddia doğrulamasının nasıl kodlandığı görülmektedir.

Daha sonra istenilen kapsama oranınan göre; örneğin rastgele 100 % düğüm ya da kenar kapsama gibi, Graphwalker modellerden test izleri üretir. Bu testler aynı zamanda içeriği doldurulmuş şablonlar ile çalıştırılır. Şekil 9'da, üretilen modeller üzerinden uygulanan bir teste ait sonuç görülmektedir. Yapılan testler için ortalama test süresi 240.06 sn'dir. Test sonuçları rastgele 100 % düğüm (random(vertex_coverage(100)) ve kenar kapsamı (random(edge_coverage(100)) kullanılarak üretilmiştir.

```
Result :
{
  "totalFailedNumberOfModels": 0,
  "totalNotExecutedNumberOfModels": 0,
  "totalNumberOfUnvisitedVertices": 0,
  "verticesNotVisited": [],
  "totalNumberOfModels": 1,
  "totalCompletedNumberOfModels": 1,
  "totalNumberOfVisitedEdges": 29,
  "totalIncompleteNumberOfModels": 0,
  "edgesNotVisited": [],
  "vertexCoverage": 100,
  "totalNumberOfEdges": 29,
  "totalNumberOfVisitedVertices": 16,
  "edgeCoverage": 100,
  "totalNumberOfVertices": 16,
  "totalNumberOfUnvisitedEdges": 0
}
```

Şekil 9. Test Sonuçları

V. Sonuç

Bu çalışmamızda model bazlı test aracı olan Graphwalker ile açık API tabanlı bir ödeme sisteminin test otomasyon süreci anlatılmıştır. Ödeme sistemi üzerinde yapılan haftalık geliştirmelerin test sunucusu üzerinde yayınlanmasını takiben sistemin uçtan uca testi sağlanır. Böylelikle geliştirilen kodlar, üretim ortamına çıkmadan önce bu çalışmamızda anlatılan test otomasyon süreci ile test edilir ve olası hatalar önceden farkedilmiş olur.

Referanslar

[1] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz "Model-based testing in practice," In Proceedings of the 21st international conference on Software engineering, pp. 285–294, 1999.

[2] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, "Mobiguitar: Automated model-based testing of mobile apps," IEEE Software, 32(5):pp.53–59, 2015.

[3] K. Moran, M. L. V´asquez, C. Bernal-C´ardenas, C. Vendome, and D. Poshyvanyk, "Automatically Discovering, Reporting and Reproducing Android Application Crashes," In IEEE International Conference on Software Testing, Verification and Validation (ICST), pp. 33–44, 2016. https://www.android-dev-tools.com/crashscope-home.

[4] K. Mao, M. Harman, and Y. Jia, "Sapienz: Multi-objective Automated Testing for Android Applications," In 25th International Symposium on Software Testing and Analysis (ISSTA), pp. 94–105, 2016.

[5] Y. Koroglu, A. Sen, O. Muslu, Y. Mete, C. Ulker, T. Tanriverdi, and Y. Donmez, "QBE: QLearning-Based Exploration of Android Applications," In IEEE International Conference on Software Testing, Verification and Validation (ICST), 2018.

[6] Y. Zhao, T. Yu, T. Su, Y. Liu, W. Zheng, J. Zhang, and W. G. J. Halfond, "Recdroid: Automatically reproducing android application crashes from bug report," In Proceedings of the 41st International Conference on Software Engineering, ICSE, pp.128–139. IEEE Press, 2019.

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

[8] Gherkin language. https://en.wikipedia.org/wiki/Cucumber (software)Gherkin language.

[9] N. Gandhewar and R. Sheikh, "Google android: An emerging software platform for mobile devices,"International Journal on Computer Science and Engineering, 1(1):pp.12–17, 2010.

[10] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon, "Using gui ripping for automated testing of android applications," In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 258–261. ACM, 2012.

[11] W. Choi, G. Necula, and K. Sen, "Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning," In ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA), pp. 623–640, 2013.

[12] "Model with ease," GraphWalker. [Online]. Available: https://graphwalker.github.io/.

[13] "WebDriver," WebDriver :: Documentation for Selenium. [Online]. Available: https://www.selenium.dev/documentation/en/webdriver/. https://github.com/GraphWalker/graphwalker-project/wiki.

.