Özellik Yönelimli Ürün Konfigürasyonlarının Olay Sıra Çizgeleri ile Doğrulanması

Tuğkan Tuğlular[1] , Fevzi Belli[2] ve Dilek Öztürk[1]

[1] Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey
tugkantuglular@iyte.edu.tr
dilekozturk@.iyte.edu.tr
[2] Department of Electrical Eng. and Information Technology, Univ. of Paderborn, Germany
fevzibelli@upb.de

Özet. Bu çalışma, özellik yönelimli yazılımlarda potansiyel olarak çok fazla sayıda olabilecek olan ürün konfigürasyonlarının, diğer bir deyişle varyantlarının, testiyle ilgili model tabanlı bir yaklaşım önermektedir. Özellik yönelimli yazılım geliştirme, yazılımın yeniden kullanımını verimli bir şekilde hayata geçirmek için kabul görmüş bir yaklaşımdır. Ürün varyantlarını yazılımın yeniden kullanımından faydalanarak geliştirirken, bu varyantları, testlerin yeniden kullanımından faydalanarak doğrulama gereksinimi doğar. Ancak bu şekilde geliştirilen yazılımların doğrulanması sürecinde testlerin yeniden kullanımı, üzerinde az çalışılmış bir konudur. Bu çalışmada, özellikleri arasında bağımlılık (gerektirme) veya çakışma (dışlama) bulunmayan özellik yönelimli yazılımların testi için en geniş kapsamdan varyanta inen, model temelli bir yaklaşım önerilmiştir. Vaka çalışmasında, değerlendirmeye tabi tutulan yazılımın modellenmesi ile hem pozitif hem de negatif test durumlarının üretilmesi için Olay Sıra Çizgeleri kullanılmıştır. Üretilmiş test durumları web test otomasyon aracı olan ve Olay Sıra Çizgelerinden otomatik olarak elde edilen betikleri işleten SahiPro ile gerçekleştirilmiştir.

Anahtar Kelimeler: Model-tabanlı Test, Olay Sıra Çizgesi, Özellik Modeli

Validation of Feature-Oriented Product Configurations Using Event Sequence Graphs

Abstract. This study attempts to suggest an approach to systematically test potentially very large number of product variants in feature-oriented software. Feature-oriented software forms a popular concept to efficiently realize software reuse. Developing feature-oriented software is well accepted to accomplish software reuse in an efficient way. Developing product variants by exploiting software reuse requires verification of these variants by exploiting test reuse. However, the reuse of tests in the verification of variants is an underworked topic. In this study, we propose a model-based approach to top-down testing of feature-oriented software that does not have dependency or conflict between features. In the case study, event sequence graphs (ESGs) are used to model the software

under consideration and then to generate test cases for positive and negative testing. The generated tests are executed via SahiPro web test automation tool, of which scripts are also automatically generated from ESGs.

Keywords: Model-based Testing, Event Sequence Graph, Feature Model

1 Giriş

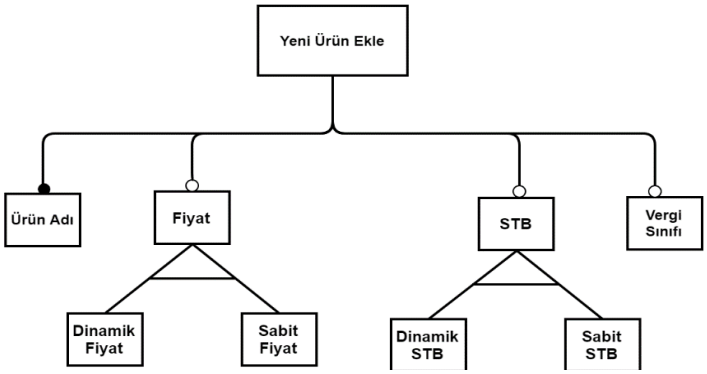
Günümüz dünyasında, bazı yazılım şirketleri ürün aileleri ya da kısmen farklı özelliklere sahip benzer ürünler üretmeyi tercih ederler. Örneğin bazı yazılım uygulamalarının standart, profesyonel ve kurumsal olmak üzere farklı konfigürasyonları bulunmaktadır. Son yıllarda, yazılımın yeniden kullanımı stratejisi, şirketler tarafından etkili ve kazançlı bir geliştirme süreci oluşturmak için kullanılmaktadır. Özellik yönelimli yazılım ürünleri, geliştirme sürecinde yazılımların sistematik olarak yeniden kullanımını mümkün kılar.

Bir özellik (İng. feature), sistem gereksinimlerini karşılayan ve potansiyel konfigürasyon seçenekleri sağlayan bir fonksiyonellik birimidir [1]. Özellikler sadece yazılımın yapısından veya davranışından meydana gelmezler: Açıkça ve sistematik bir biçimde değişkenlikleri ve ortaklıkları tanımlarlar ve yazılımın yeniden kullanımına olanak sağlarlar [1].

Özellik yönelimli yazılımlar (İng. feature-oriented software) belirli bir sektörün ihtiyaçlarını karşılamak için tasarlanmıştır. Çoğunluğu ortak özellikler olmak üzere farklı özellikleri de bünyesine katabilen yazılım ürünlerini içerir. Yazılımın yeniden kullanımı ürünü en baştan geliştirmek yerine var olan yazılımı geliştirmek veya genişletmektir. Özellik yönelimli yazılımları geliştirirken, yazılımın yeniden kullanımından faydalanılması düşük maliyetli, daha az zaman ve emek gerektiren, daha kaliteli ürünler elde etmemizi sağlayan, etkili ve üretken bir geliştirme süreci ortaya koyar.

Yazılımın yeniden kullanımında ürünlerin özellik değişimlerini desteklemesi gerekir ancak, değişim arttıkça geliştirme ve test etme sürecinin karmaşıklığı artar [1]. Karmaşıklığı artırmak yazılımın yeniden kullanımını riskli bir hale getirebilir. Bir bileşen bir varyanta mükemmel uyum sağlarken, farklı bir varyant bünyesinde ciddi hatalar ortaya çıkarabilir. Varyantların doğrulama ve sağlamasının yapılması zorlu bir görevdir, ancak ürün kümesindeki her varyantın kaliteli olduğunu garantilemek için gereklidir.

Özellik yönelimli yazılımları test etmek için birçok yaklaşım bulunmaktadır. Bunlardan biri, kaba kuvvet (İng. brute force) yaklaşımıdır. Bu yaklaşımı takip ederek, ürünleri tek tek test etmek pek uygulanabilir değildir, çünkü konfigürasyonların sayısı çok fazla olabilir.Bir özellik kümesinden, ortak ve farklı özelliklere sahip olan birçok yazılım üretilebilir [1]. Özellik kümeleri kullanılarak yazılım üretim süreci yazılım ürün hattı (İng. software product line) yaklaşımı olarak karşımıza çıkmaktadır [1].



Şekil 1. Magento Yeni Ürün Ekle Özellik Modeli

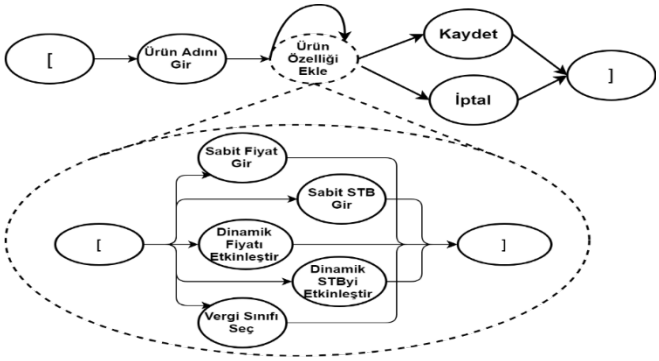
Bu çalışmada, özellikleri arasında bağımlılık (gerektirme) veya çakışma (dışlama) bulunmayan özellik yönelimli yazılımların ürün kümelerinde bulunan her bir spesifik ürünün kalitesini iyileştirmek için, en geniş kapsamdan varyanta inen, model temelli yeni bir yaklaşım ortaya konulmuştur. İzleyen bölümde önerilen yaklaşım açıklanmıştır. Bir sonraki bölümde vaka çalışması ortaya konmuş ve sonra da elde edilen sonuçlar tartışılmıştır. Yaklaşım, vaka çalışması ve tartışma bölümlerinin ardından ilgili çalışmalara yer verilmiştir. Sonuç bölümünde önerilen yaklaşımın önemi vurgulanmıştır.

2 Önerilen Yaklaşım

Bu bildiride kullanılan vaka çalışması için ürün konfigürasyonu değişim noktalarını gösteren bir özellik modeli oluşturulmuştur ve Şekil 1'de gösterilmiştir. Öncelikle, Şekil 1'deki özellik modelinin tamamını kapsayan Şekil 2'deki Olay Sıra Çizgesi (İng. Event Sequence Graph, kısaca OSÇ) eşleştirilmiş ve bu eşleştirmenin sonucunda, olası bir ürünü temsil eden bir varyant Olay Sıra Çizgesi türetilmiş ve EK-A'da yer alan şekillerde gösterilmiştir. Her bir varyant Olay Sıra Çizgesi, ilgili ürün varyantının test durumlarını üretmek için kullanılmıştır.

Araştırmada kullanılan vaka çalışmasındaki örnekler, Magento [2] isimli gerçek bir yazılımından alınmıştır. Magento özellik yönelimli yazılımlara benzerlik gösteren modül ve fonksiyonelliklere sahip bir e-ticaret yazılımıdır. İngilizce olarak kullanımda olan bu yazılımın çeşitli modülleri ve kullanıcı olayları bu çalışmada Türkçe 'ye çevrilmiş ve öyle sunulmuştur.

Üzerinde çalışılan özellik yönelimli yazılıma ait özellik modeli Şekil 1'de verilmiştir. En üstteki özellik örneğimizdeki ürün kümesinin adıdır. Bu isim Magento'daki Yeni Ürün Ekle (İng. Add New Product) isimli özelliğe karşılık gelmektedir. Bir Magento kullanıcısı, bu olayı gerçekleştirdiğinde yeni bir ürün, ürün kataloğuna eklenmektedir. Eklenen ürün Basit (İng. Simple), Konfigüre-Edilebilir (İng. Configurable), Paket (İng. Bundle), Gruplanmış (İng. Grouped), Sanal (İng. Virtual) ve İndirilebilir (İng. Downloadable) isimli ürün varyantlarından biri olabilir. Bu altı ürün bir arada, bir özellik yönelimli yazılım ürün kümesi oluşturmaktadır.



Şekil 2. Kapsayan Olay Sıra Çizgesi.

Verilen özellik modelinde, Ürün Adı (İng. Product Name), Fiyat (İng. Price), Stok Tutma Birimi (İng. Stock Keeping Unit, kısaca STB) ve Vergi Sınıfı (İng. Tax Class) isimli dört özellik en yukarıdaki özelliğe bağlanmıştır. STB, ürünleri eşsiz olarak tanımlamaya yarayan kodun kısaltılmasıdır [3]. Ürün Adı kutusunun üstünde bulunan içi dolu halka bu özelliğin her üründe mutlaka bulunduğu anlamına gelmektedir. İçi boş halkalarla biten Fiyat, STB ve Vergi Sınıfı seçmeli özelliklerdir. Fiyat ve STB birbirine alternatif olan iki alt-özellik bulundurmaktadır. Bir ürün varyantı için alternatif özelliklerden sadece bir tanesi seçilebilir [4].

Magento'da ürün kataloğuna yeni bir ürün eklemek isteyen bir kullanıcı, bir ürün varyantı seçer ve bu ürünün tipine göre ürünle ilgili bazı bilgiler doldurur. Kataloğa eklenmek istenen bütün ürünler için Ürün Adı zorunlu iken, Fiyat, SKU ve Vergi Sınıfı belirtilmeden bırakılabilir.

Olay Sıra Çizgesi bir olay sırasını (İng. event sequence) temsil eder [5]. Uzunluğu iki olan olay sıraları, olay çiftleri (İng. event pairs) olarak adlandırılır. Bir olay sırası başlangıç ve bitiş düğümlerini içeriyorsa, tam olay sırası (İng. complete event sequence) veya legal yürüyüş (İng. legal walk) olarak adlandırılır [6]. Gerekli uzunluktaki olay sıralarını kaplayan, minimal toplam uzunluktaki tamamlanmış olay sırası kümelerinin birleşimi, minimal bir test takımını (İng. test suite) temsil eder. Bir tamamlanmış olay sırası, bütün olay çiftlerini en az bir kez içeriyorsa, tam yürüyüş (İng. entire walk) olarak adlandırılır [6]. Bir legal tam yürüyüş, eğer uzunluğu azaltılamıyorsa minimaldir. Minimal bir legal yürüyüş, olay çiftlerini tam olarak bir kez içeriyorsa idealdir [6].

Legal yürüyüşler, verilmiş bir Olay Sıra Çizgesinden kolaylıkla üretilebilir. Ancak, tam yürüyüş ya da ideal yürüyüş oluşturmak her zaman mümkün olmayabilir. Çizge kuramının bazı kuramları kullanılarak [7], minimal test takımları oluşturulabilir [6]. İlgili algoritmalar ve uygulamaları [5], [8] ve [9]'de görülebilir.

Hatalı (ya da legal olmayan) bir olay çifti (İng. faulty event pair) herhangi bir olay ile başlar ve verilmiş Olay Sıra Çizgesi ile uyumluluk göstermeyen ve verilen olayı takip etmemesi gereken bir olay ile devam eder [5]. Tam hatalı olay çifti (İng. complete

Tablo 1.Ürün Konfigürasyonlarının Özellik Seçimi.

	Basit Ürün	Paket Ürün	İndirilebilir Ürün
Ürün Adı	✓	✓	✓
Sabit SBT	✓		✓
Dinamik SBT		✓	
Sabit Fiyat	✓		
Dinamik Fiyat		✓	✓
Vergi Sınıfı	✓		✓

faulty event sequence), sistemi hatalı bir duruma getirmesi gereken bir test durumunu temsil eder [5]. Tamamlanmış olay sıraları ile ilgili terminoloji, tamamlanmış hatalı olay sıralarını göz önünde bulundurarak genişletilebilir.

Tamamlanmış olay sıraları pozitif testleri gerçekleştirmek için kullanılırken, tamamlanmış hatalı olay sıraları negatif testleri gerçekleştirmekte kullanılır [5]. Pozitif ve negatif test durumlarının nasıl üretildiği izleyen bölümde açıklanmıştır.

3 Vaka Çalışması

Vaka çalışması, Magento Community Edition 2.1.2 sürümü kullanılarak aşağıda yer alan basitleştirmelerle oluşturulmuştur. Söz konusu Magento sürümünün Yeni Ürün Ekle özellik modelinde on üç özellik bulunurken, bizim modelimizde bu özelliklerin yalnızca dört tanesi seçilmiştir. Basit, Konfigüre-Edilebilir, Gruplanmış, Paket, Sanal ve İndirilebilir isimli altı farklı varyant arasından sadece Basit, Paket ve İndirilebilir isimli toplam üç varyant seçilmiştir. Bu ürün varyantları birlikte özellik yönelimli bir yazılımı temsil eder ve aralarında bağımlılık (gerektirme) veya çakışma (dışlama) bulunmamaktadır. Vaka çalışmasında yer alan her varyant, doldurulacak ortak ve farklı alanlar gerektirir. Tablo 1 varyant ürünler için doldurulması gereken ortak ve farklı alanları göstermektedir.

Tablo 1'de verilen varyant özellikleri kullanılarak, Basit Ürün, Paket Ürün ve İndirilebilir Ürün için oluşturulan varyant OSÇleri sırasıyla Şekil 3, Şekil 4 ve Şekil 5'de gösterilmiştir. Daha sonra her bir OSÇ için, tam olay sıraları ve tam hatalı olay sıraları Test Suite Designer (kısaca TSD) kullanılarak oluşturulmuştur.

Paderborn Üniversitesi tarafından geliştirilen ve araştırmalar için ücretsiz olarak kullanılabilen TSD aracı Olay Sıra Çizgelerini kullanarak otomatik test durumları oluşturmaktadır [10].

TSD aracı Almanca dilindedir. Aracı çalıştırmak için gerekli dosya kaynakça [10]'da verilen bağlantıdaki web sayfasından indirilebilir. İndirilen dosya, çalıştırılabilir ve ".jar" uzantılı bir dosyadır. Bu dosya çalıştırıldığı zaman model çizim arayüzü ekranda belirir. Bir OSÇ çizildikten sonra ".mxe" uzantılı bir dosya olarak kaydedilir. TSD arayüzünde bulunan "Generiere Test-Skripte" isimli butona tıklayarak çizilmiş olan OSÇ'ye ait pozitif ve negatif test durumları elde edilir. Elde edilen test durumları yeni bir pencerede gösterilir ve istenirse metin dosyası olarak kaydedilebilir.

Tablo 2. Dört OSÇ için Olay Sıra Sayısı ve Olay Sayısı Karşılaştırması

	Uzunluk 2	**Olay Sıra Sayısı**	**Olay Sayısı**
OSÇ 1 Basit Ürün	*Tam Olay Sırası*	6	27
	Hatalı Tam Olay Sıra Sayısı	23	68
OSÇ 2 Paket Ürün	*Tam Olay Sırası*	4	16
	Hatalı Tam Olay Sıra Sayısı	19	56
OSÇ 3 İndirilebilir Ürün	*Tam Olay Sırası*	6	27
	Hatalı Tam Olay Sıra Sayısı	23	68
OSÇ 4 Kapsayan OSÇ	*Tam Olay Sırası*	10	55
	Hatalı Tam Olay Sıra Sayısı	31	92

TSD kullanılarak, OSÇ'de bulunan her bir düğüme o düğümle ilgili test betikleri kaydedilebilir ve test durumları otomatik olarak elde edilirken, bütün ürün varyantını kapsayacak bir test betiği de istenilen uzantıda (.html, .xml vb.) otomatik olarak üretilir. Çalışmamızda SahiPro isimli ".sah" uzantılı dosyaları otomatik olarak çalıştıran test programı kullanıldığı için, uzantı ".sah" olarak seçilmiştir.

Basit Ürün varyantı için üretilen ve olay çiftlerini kapsayan 6 tam olay sırası aşağıda verilmiştir. Örneğin <ÜrünAdıGir, SabitSBTGir>, <SabitSBTGir, Kaydet> ve <VergiSınıfıSeç, İptal> tam olay çiftleridir.

12: [, ÜrünAdıGir, SabitSBTGir, SabitSBTGir, SabitFiyatGir, SabitFiyatGir, VergiSınıfıSeç, VergiSınıfıSeç, SabitFiyatGir, SabitSBTGir, VergiSınıfıSeç, SabitSBTGir, Kaydet,]
3: [, ÜrünAdıGir, SabitFiyatGir, Kaydet,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, Kaydet,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, İptal,]
3: [, ÜrünAdıGir, SabitSBTGir, İptal,]
3: [, ÜrünAdıGir, SabitFiyatGir, İptal,]

Basit Ürün varyantı için 23 tam hatalı olay sırası üretilmiştir. Test durumlarından önceki sayılar, test durumunun uzunluğunu verir. Uzunluklar hayali düğümler ([,]) hariç tutularak ölçülür.

Paket Ürün varyantı için üretilen ve olay çiftlerinin kapsayan 4 tam olay sırası aşağıda verilmiştir. Paket Ürün varyantı için 19 hatalı olay sırası üretilmiştir.

7: [, ÜrünAdıGir, DinamikSBTyiEtkinleştir, DinamikSBTyiEtkinleştir, DinamikFiyatıEtkinleştir, DinamikFiyatıEtkinleştir, DinamikSBTyiEtkinleştir, Kaydet,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, Kaydet,]
3: [, ÜrünAdıGir, DinamikSBTyiEtkinleştir, İptal,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, İptal,]

İndirilebilir Ürün varyantı için üretilen ve olay çiftlerini kapsayan 6 tam olay sırası aşağıda verilmiştir. İndirilebilir Ürün varyantı için 23 hatalı olay sırası üretilmiştir.

12: [, ÜrünAdıGir, SabitSBTGir, SabitSBTGir, DinamikFiyatıEtkinleştir, DinamikFiyatıEtkinleştir, VergiSınıfıSeç, VergiSınıfıSeç, DinamikFiyatıEtkinleştir, SabitSBTGir, VergiSınıfıSeç, SabitSBTGir, Kaydet,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, Kaydet,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, Kaydet,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, İptal,]
3: [, ÜrünAdıGir, SabitSBTGir, İptal,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, İptal,]

Tablo 2'de Basit, Paket ve İndirilebilir ürün varyantları ve bu ürün varyantlarına ait kapsayan OSÇ'nin doğru ve hatalı test durumları bakımından olay sıra ve olay sayıları verilmiştir. Bu sayılar, uzunluğu iki olan olay sıraları yani olay çiftleri için TSD tarafından hesaplanmıştır. Basit Ürün için 6 tam olay sırası ve 27 olay, 23 hatalı olay sırası ve 16 olay; Paket Ürün için 4 tam olay sırası ve 16 olay, 19 hatalı tam olay sırası ve 56 olay; İndirilebilir Ürün için 10 tam olay sırası ve 55 olay, 31 hatalı tam olay sırası ve 92 olay bulunmaktadır. Kapsayan OSÇ'nin 10 tam olay sırası ve 55 olayı, 31 hatalı tam olay sırası ve 92 olayı bulunmaktadır ve görüldüğü gibi, varyantların her birinden daha fazla olay ve olay sırası içermektedir. Bu durum kapsayan OSÇ'nin, Magento'nun tarafımızdan seçilen bütün özellikleri bulundurmasından kaynaklanmaktadır.

Bu çalışmada, ürün varyantları için otomatik bir biçimde test durumu ve test betiği üretmek için TSD'den yararlanılmıştır. Test betiklerini otomatik olarak çalıştırmak için ise bir test otomasyon aracı olan SahiPro kullanılmıştır. SahiPro, web ürünleri üzerinde kaydet ve yeniden oynat (İng. record and playback) prensibini kullanarak test betikleri üretir ve bunları çalıştırır. Ayrıca, başka bir ortamda üretilmiş betikleri de çalıştırabilir ve otomatik bir biçimde test yapılmasını sağlar.

SahiPro'nun kaydet ve yeniden oynat özelliği kullanılarak, Magento'daki kullanıcı olaylarına, yani OSÇ'lerdeki düşümlere karşılık gelen test betikleri elde edilmiştir. Her bir düğüm için ayrı ayrı edilen bu betikler, TSD'de çizilen OSÇ'lerin düğümlerine kaydedilmiş ve test durumu üretimi sırasında otomatik olarak elde edilmiştir. Elde edilen bu betik Magento üzerinde SahiPro kullanılarak otomatik olarak çalıştırılmış ve test gerçekleştirilmiştir.

4 Tartışma

Kapsayan OSÇ için üretilen ve olay çiftlerinin kapsandığı 10 tam olay sırası aşağıda verilmiştir. Kapsayan OSÇ için 31 hatalı tam olay sırası üretilmiştir.

28: [, ÜrünAdıGir, SabitSBTGir, SabitSBTGir, SabitFiyatGir, SabitFiyatGir, VergiSınıfıSeç, VergiSınıfıSeç, DinamikFiyatıEtkinleştir, DinamikFiyatıEtkinleştir, DinamikSBTyiEtkinleştir, DinamikSBTyiEtkinleştir, DinamikFiyatıEtkinleştir, VergiSını-

fıSeç, DinamikSBTyiEtkinleştir, VergiSınıfıSeç, SabitFiyatGir, DinamikFiyatıEtkinleştir, SabitFiyatGir, DinamikSBTyiEtkinleştir, SabitFiyatGir, SabitSBTGir, VergiSınıfıSeç, SabitSBTGir, DinamikFiyatıEtkinleştir, SabitSBTGir, DinamikSBTyiEtkinleştir, SabitSBTGir, Kaydet,]

3: [, ÜrünAdıGir, SabitFiyatGir, Kaydet,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, Kaydet,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, Kaydet,]
3: [, ÜrünAdıGir, DinamikSBTyiEtkinleştir, Kaydet,]
3: [, ÜrünAdıGir, DinamikSBTyiEtkinleştir, İptal,]
3: [, ÜrünAdıGir, DinamikFiyatıEtkinleştir, İptal,]
3: [, ÜrünAdıGir, VergiSınıfıSeç, İptal,]
3: [, ÜrünAdıGir, SabitSBTGir, İptal,]
3: [, ÜrünAdıGir, SabitFiyatGir, İptal,]

Üç varyant ürünün tam olay sıraları ile kapsayan OSÇ'den elde edilen 10 tam olay sırası karşılaştırıldığında, hariç tutulan özellikleri içeren olay çiftlerinin, buna karşılık gelen varyantın tam olay sıralarında bulunmadığı gözlemlenmiştir. Örneğin, Sabit SBT, Sabit Fiyat ve Vergi Sınıfı özelikleri bulunmayan Paket Ürün'ün test takımında bu özellikleri içeren herhangi bir olay çifti bulunmamaktadır. Benzer şekilde, İndirilebilir Ürün'ün sahip olmadığı Dinamik SBT ve Sabit Fiyat özelliklerine ait olay çiftleri, bu varyantın test takımında bulunmamaktadır.

Hariç tutulan özellikleri kapsayan olay çiftlerini bulundurmamanın yanı sıra, dahil edilen özellikler bakımından, kapsayan OSÇ'ye ait test takımı ile varyant ürünlerin test takımları arasında ortaklıklar bulunmaktadır. Örneğin, Basit Ürün'e ait 6 tam olay sırasının 5 tanesi Kapsayan OSÇ'nin test takımında mevcuttur. Benzer biçimde, Paket ürüne ait 4 tam olay sırasının 3 tanesi Kapsayan OSÇ ile ortaktır.

Bu gözlemlerden yola çıkılarak, önerilen yaklaşımda, bir varyant için hariç tutulan özellikleri kullanarak, kapsayan OSÇ'nin tam olay sıralarından, varyant ürünlerin tam olay sıralarının üretilmesinin mümkün olabileceği sonucu çıkarılmıştır. Varyant ürünlerle ortak olan özellikleri kapsayan olay çiftlerini içeren tam olay sıraları, kapsayan OSÇ'nin test takımı oluşturulduğu anda elde edilmiş durumdadır. Sayfa sınırlamalarından dolayı 4 olay sıra çizgesinin hatalı tam olay sıraları verilememiş olsa da yukarıda açıklanan durumun hatalı tam olay sıraları için de benzer olduğunu gözlemlenmiştir.

5 İlgili Çalışmalar

Model tabanlı test yöntemlerini, özellik modellerini ve özellik yönelimli yazılımları ön planda tutan bu çalışmada, literatür taraması bu konular ve yazılım ürün hatları göz önünde bulundurularak yapılmıştır.

Özellik yönelimli yazılımlar, yazılım ürün hatları veya yazılım ürün ailelerine çeşitli yönlerden benzerlik göstermektedir. İki yaklaşımın da yazılımın yeniden kullanımına uygunluk göstermesi, değişkenlik noktaları sayesinde çok fazla sayıda varyant üretimini sağlayabilmesi, alan mühendisliği (İng. domain engineering) ve uygulama mühendisliği (İng. application engineering) süreçleri ile geliştirmeye açık olması benzer

yönlerden bazılarıdır. Yazılım ürün hatları ortak bir yazılım düzlemini taban alır ve bu düzlem üzerinde uyarlamalar yapılarak birçok spesifik ürün elde edilmesine olanak sağlar [11]. Bir yazılım ürün hattını tam olarak test edebilmek için, ortak yazılım düzleminin ve olası bütün ürün varyantlarının test edilmesi gerekmektedir [11]. Özellik yönelimli yazılımlara benzer şekilde, yazılım ürün hatlarından elde edilen varyant ürün sayısı değişkenlik noktalarıyla birlikte katlanarak arttığı için, tekrar eden testlerin sayısını ve test için harcanan efor ve maliyeti düşürmek elzem bir hal almaktadır. Özellik yönelimli yazılımlar ve yazılım ürün hatları, benzer yönlere ve benzer sorunlara sahip oldukları için, literatür taramasına yazılım ürün hatları ile ilgili olan yayınlar dahil edilmiştir.

2011 yılında yazılım ürün hatlarındaki test süreçlerini inceleyen iki literatür tarama çalışması [11-12] yapılmıştır. Model tabanlı test üretim sürecini adreslemeyen iki çalışmadan birincisi [11] yazılım ürün hatlarını test ederken karşılaşılan büyük sorunlar olan çok fazla sayıda test durumu, tekrar kullanılabilir olan bileşenlerin testi ve değişkenlikten bahsetmiştir. İkinci literatür tarama çalışması ise [12] yazılım ürün hatlarıyla var olan test stratejilerinin, statik ve dinamik analiz yöntemlerinin, test seviyelerinin, fonksiyonel olmayan gereksinimlere ve regresyon testine olan yaklaşımların, ortaklık ve değişkenlikle ilgili çıkabilecek sorunların ve harcanan test eforunun ilişkisine odaklanmıştır.

Yazılım ürün hatlarının model tabanlı testi için 2004 yılında yayınlanan bir çalışmada [13] modelin sınırlı durum makinesi (İng. finite-state machine) olması ve ürünün model denetleme ile gereksinimlerde belirlenen özelliklere sahip olup olmadığının araştırılması önerilmiştir.

2005 yılında yapılan ve yazılım ürün ailelerinin model tabanlı sistem testine odaklanan bir çalışmada [14] değişkenliği modelleyerek, yazılımın yeniden kullanımına olanak sağlandığı ve bu sayede geliştirme maliyetlerinin ve ürünün piyasaya sürümü için geçen zamanın azaldığı belirtilmiştir. Yazılımın yeniden kullanımının faydalarından test süreçleri düşünülmeyerek, sadece geliştirme süreçlerinde yararlanıldığı anlatılan bu çalışma ScenTED (Scenario based Test case Derivation) isimli, ürün ailelerini test etmek için gereken eforun düşürülmesini amaçlayan bir teknik ortaya koymuştur. UML kullanım durumlarından (İng. use case), UML etkinlik diyagramlarını (İng. activity diagram) üreten ve bu etkinlik diyagramlarını test üretici model olarak kullanan teknik, aynı zamanda hiyerarşik etkinlik diyagramlarını da ileri sürüp, kullanmıştır.

2008 yılında yazılım ürün hatlarının model tabanlı testleri konusunda yapılan bir başka çalışmada [15] CADeT (Customizable Activity Diagrams, Decision Tables and Test Specifications) isimli bir yöntem sunulmuştur. Özellik bazlı test durumları üreten, değişkenlikleri modellerken ve testleri üretirken karar tablolarından (İng. decision table) faydalanan bu yöntem, UML kullanım durumlarını ve UML etkinlik diyagramlarını kullanır.

2011 yılında yapılan bir çalışmada [16] test üretici model sınırlı durum makineleridir. Temel bileşenlere ait sınırlı durum makinesi ile özellik modelleri birleştirilerek varyant ürünleri temsil eden yeni sınırlı durum makineleri ve bunlara ait test durumları oluşturulmuştur. 2013 yılında yapılan bir çalışmada [4] da test üretici model sınırlı durum makinesi olup, özellik modelleri kullanılarak değişkenlik noktaları belirlenmiştir. Bütün ürün hattının özelliklerini kapsayan %150 sınırlı durum makinesi ve varyant

ürünlerin özelliklerini kapsayan %100 durum makinesi oluşturulmuştur. %100 ve %150 sınırlı durum makineleri ve özellik modelleri kullanılarak ve çeşitli kapsam kriterleri belirlenerek yukarıdan-aşağı ve aşağıdan-yukarı test yöntemleri ile test durumları oluşturulmuştur.

2004 yılında yapılan bir çalışmada [17], değişkenlik noktalarının, test mühendislerine nereleri daha iyi test edecekleri konusunda fikir verdiğini ve geliştirme ile test süreçlerinde aynı değişkenlik mekanizmasının kullanılmasını önermiştir.

Çalışmamızda kullanılan TSD aracına benzer şekilde yönlü çizgelerden test durumları üreten GraphWalker [18] isimli bir araç da bulunmaktadır. GraphWalker çeşitli matematiksel algoritmalar sayesinde verilen model ve üretici kuralı kullanarak test durumları oluşturur. Araçta kullanılan test modeli olan yönlü çizgedeki kenarlar eylemleri, düğümler ise doğrulamayı temsil etmektedir.

6 Sonuç

Günümüzde, benzer ürünleri olan veya hedefleyen kuruluşların üretim stratejisi, ürünleri sıfırdan geliştirmek yerine yazılımın yeniden kullanımı tekniğine odaklanarak ortak özelliklere sahip yazılım ürünleri geliştirmektedir. Bu sayede yazılım ürünleri verimli ve karlı şekilde geliştirilebilmektedir. Ortak kanıya göre, yazılımın yeniden kullanım stratejisi ile özellik yönelimli ürün konfigürasyonlarının iyi bir kombinasyonu, nitelikli bireysel ürünler, maliyet-, emek- ve zaman- dostu bir geliştirme süreci vadetmektedir.

Özellik yönelimli yazılımlar, ürün konfigürasyonlarını üretmektedir. Ancak, üretilen konfigürasyon sayısı arttıkça, geliştirme ve test etme karmaşıklığı da artmaktadır. Yeniden kullanılan bileşenlerinin ürün veya sistemlere olası uyumsuzluğundan dolayı bu karmaşıklık arttığında yazılımın yeniden kullanımı riskli olmaktadır. Üretkenliği ve kaliteyi artırdığı, maliyet, zaman ve işgücü ihtiyacını azalttığı için bir ürün ailesi tanımlamak son derece makul ve avantajlıdır, ancak yazılımın yeniden kullanımının doğası gereği her ürünü tek tek doğrulamak gerekir.

Bu bildiride, özellik yönelimli ürün konfigürasyonları için test durumlarını otomatik ve sistematik bir şekilde oluşturmaya yönelik yeni bir yaklaşım sunulmuştur. Bu çalışmada, özellikleri arasında bağımlılık (gerektirme) veya çakışma (dışlama) bulunmayan özellik yönelimli yazılımların testi için en geniş kapsamdan varyanta inen, model temelli bir yaklaşım önerilmiştir. Bu çalışma boyunca ürün değişimlerini modellemek için özellik modelleri ve Olay Sıra Çizgeleri kullanılmıştır.

Gelecek araştırma hedeflerimizden biri, özellik yönelimli ürün konfigürasyonları için otomatik test ortamları oluşturmak amacıyla özellik modellerini Olay Sıra Çizgeleri ile otomatik olarak eşleştirmektir.

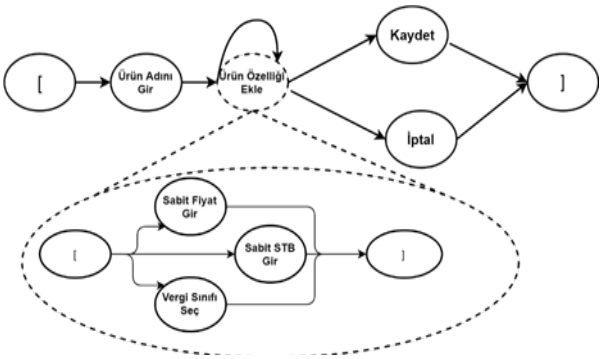
Teşekkür

Bu çalışma, TÜBİTAK tarafından 117E884 numaralı "Çevik Yazılım Ürün Hatları için Olay Sıra Çizge Tabanlı Test Üretim Yöntemi Geliştirilmesi" başlıklı proje kapsamında desteklenmiştir, teşekkürlerimizi sunarız.

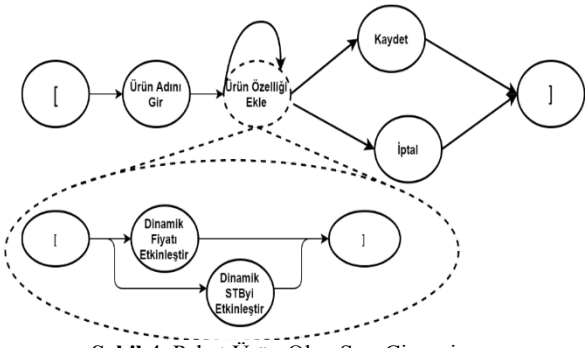
Kaynakça

1. Apel, S. Kästner, C. An Overview of Feature-Oriented Software Development. (2009). in Journal of Object Technology, vol. 8, no. 5, July–August 2009, pp. 49–84.
2. Magento. (n.d). https://magento.com/products/community-edition. 09.08.2018 tarihinde erişilmiştir.
3. The balance: What Is a Stock Keeping Unit (SKU)?. https://www.thebalance.com/what-is-a-sku-in-retail-terms-2890158. 09.08.2018 tarihinde erişilmiştir.
4. Weißleder, S. Lackner, H. (2013). Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines. MBT, volume 111 of EPTCS, page 82-94. Electronic Proceedings in Theoretical Computer Science (EPTCS) 111, pp. 82-94.
5. Belli, F. "Finite state testing and analysis of graphical user interfaces," in Proceedings of 12th International Symposium on Software Reliability Engineering, 2001, pp. 34–43.
6. Belli, F., Budnik, Ch. J., White, L. (2006). Event-based Modeling, Analysis and Testing of User Interactions: Approach and Case Study, Software Testing, Verification and Reliability, vol. 16, 1, 3-32, John Wiley & Sons, Ltd.
7. West, DB. (1996). Introduction to Graph Theory, Prentice Hall.
8. Belli, F. "A holistic view for finite-state modeling and testing of user interactions". (2003). Proceedings of the 1st South-East European Workshop on Formal Methods.
9. Belli, F., Linschulte, M., Tuğlular, T. (2016). Karar Tablosu Destekli Olay Sıra Çizgeleri Temelli Sınama Durum Üretim Aracı, Ulusal Yazılım Mühendisliği Sempozyumu, İzmir.
10. TestSuiteDesigner. (n.d, Univ. of Paderborn, Angew. Datemtechnik). http://download.ivknet.de/. 09.08.2018 tarihinde erişilmiştir.
11. Engström, E. Runeson, P. Software Product Line Testing- A Systematic Mapping Study. (2011). Information and Software Technology, Volume 53, Issue 1, 2011, pp. 2-13.
12. Neto, P.A.D.M.S., do Carmo Machado, I., McGregor, J.D., De Almeida, E.S., de Lemos Meira, S.R. (2011). A systematic mapping study of software product lines testing. Information and Software Technology, 53(5), 407-423.
13. Kishi, T., Noda, N. (2004). Design testing for product line development based on test scenarios. In Proc. of the international workshop on software product line testing, pp. 19-26.
14. Reuys, A., Kamsties, E., Pohl, K., Reis, S. (2005). Model-based system testing of software product families Advanced Information Systems Engineering, pp. 379-380, Springer.
15. Olimpiew, E.M. (2008). Model-based Testing for Software Product Lines. Ph.D. Dissertation, George Mason University, USA.
16. Cichos, H., Oster, S., Lochau, M., Schürr, A. (2011). Model-based coverage-driven test suite generation for software product lines. International Conference on Model Driven Engineering Languages and Systems, pp. 425-439, Springer.
17. McGregor, J.D., Sodhani, P., Madhavapeddi, S. (2004). Testing Variability in a Software Product Line. Software Product Line Testing Workshop (SPLiT), Boston, MA, pp. 45-50.
18. GraphWalker. https://graphwalker.github.io/. 09.08.2018 tarihinde erişilmiştir.

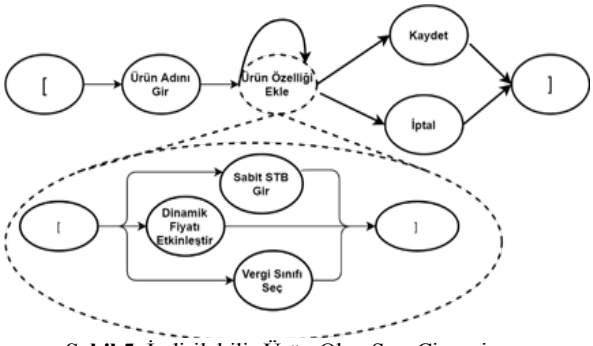
EK A. Varyant Olay Sıra Çizgeleri



Şekil 3. Basit Ürün Olay Sıra Çizgesi.



Şekil 4. Paket Ürün Olay Sıra Çizgesi.



Şekil 5. İndirilebilir Ürün Olay Sıra Çizgesi.