DEÜ FMD 25(73), 217-238, 2023

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering

Basılı/Printed ISSN: 1302-9304. Elektronik/Online ISSN: 2547-958X

Yazılım Ürün Hatlarında Tam Üründen Özellik Eksiltme Yoluyla Farklı Ürün Yapılandırmalarını Otomatik Üretme Yöntemi

A Method to Automatically Generate Different Product Configurations by Deducting Features from the Full Product in Software Product Lines

Dilek Öztürk Kaya [1*], Tuğkan Tuğlular [1]

[1] İzmir Yüksek Teknoloji Enstitüsü, Mühendislik Fakültesi, Bilgisayar Mühendisliği, İzmir, TÜRKİYE
Sorumlu Yazar / Corresponding Author *: dilekozturk@iyte.edu.tr

Geliş Tarihi / Received: 09.06.2022 *Araştırma Makalesi/Research Article*
Kabul Tarihi / Accepted: 15.08.2022 *DOI:10.21205/deufmd.2023257318*
Atıf şekli/ How to cite: ÖZTÜRK-KAYA, D., TUGLULAR, T.(2023). Yazılım Ürün Hatlarında Tam Üründen Özellik Eksiltme Yoluyla Farklı Ürün Yapılandırmalarını Otomatik Üretme Yöntemi. DEÜ FMD 25(73), 217-238.

Öz

Yazılım ürün hattı (YÜH) karmaşık, büyük ölçekli ve ürün yapılandırması bakımından zengin yazılım sistemleri geliştirmek için gelelecek vadeden bir yaklaşımdır. Yazılım ürün hattındaki sayısı çok fazla olabilen ürün yapılandırmalarına ait modellerin otomatik elde edilmesi zaman ve maliyet kısıtları açısından oldukça önemlidir. Bu çalışmada, ürün modellerini daha üretken ve etkili şekilde elde edebilmek için, tam ürün modelinden, özellik eksiltme yoluyla farklı ürün yapılandırmalarına ait modelleri otomatik olarak elde etmeyi sağlayan bir yaklaşım önerdik. Önerilen yaklaşımı İçecek Otomatı YÜH, Banka Hesabı YÜH ve Öğrenci Yoklama Sistemi YÜH isimli üç farklı vaka çalışması üzerinde denedik. Özellik-bağımlılık ağacı ve dinamik kenar eşleme algoritması bu çalışmada önerilen özgün kavramlardır.

Anahtar Kelimeler: *Yazılım Ürün Hattı, Olay Sıra Çizgesi, Eleme Koşulu, Özellik Modeli, Ürün Yapılandırması*

Abstract

The software product line (SPL) is a promising approach to developing software systems that are complex, large-scale, and rich in product configuration. The automatic acquisition of product configurations' models, which may be too many in the software product line, is very important in terms of time and cost constraints. In this study, we propose an approach that automatically extracts models of different product configurations from the full product model by feature deduction to obtain product models more productively and effectively. We validated the proposed approach on three different case studies: Soda Vending Machine SPL, Bank Account SPL, and Student Attendance System SPL. The two novelties of this study are the feature-dependency tree and the dynamic edge mapping algorithm.

Keywords: *Software Product Line, Event Sequence Graph, Elimination Condition, Feature Model, Product Configuration*

1. Giriş

Son yıllarda artan ürün çeşitliliği sebebiyle, geliştirme maliyetlerini ve sürelerini düşürürken, ürün kalitesini ve yeniden kullanım oranını iyileştirmek, geliştirme sürecinin temel hedeflerinden biridir [1]. Yazılım ürün hattı paradigması, daha kısa sürede ve daha düşük maliyet ile daha yüksek kaliteli yazılım ürünlerini geliştirebilmeyi sağlar [1]. Bir yazılım ürün hattında bulunan ürünler, belirli bir alandaki (İng. domain) birbiriyle ilişkili ortak özelliklere ve bazı ürünlerde bulunurken diğerlerinde bulunmayan, isteğe bağlı özelliklere sahiptir [2].

Bir yazılım sisteminin, kullanıcıları direkt olarak etkileyen, belirgin ya da ayırt edici karakteristikleri, ilgili sisteme ait özelliklerdir. [3]. Yazılım ürün hatları, ürün yapılandırmasına izin veren değişkenlik noktalarına sahip bir küme özellik etrafında inşa edilir [4]. Değişkenlik noktalarına sahip özellikler, YÜH'te bulunan ürünler için isteğe bağlı olan özelliklerdir [4]. Özellikler ve bunların arasındaki ilişkiler, özellik modelleri kullanılarak temsil edilir [3]. Yazılım ürün hattındaki tüm ürünlerde bulunan zorunlu özellikler ve ürünlerde bulunmada değişkenlik gösteren isteğe bağlı özellikler, özellik modelleri ile gösterilmektedir [3,5] . Özellik modellerinde, özellikler arasında VEYA ya da DIŞLAYAN VEYA ilişkileri ile birlikte gerektirme, dışlama gibi çeşitli kısıtlar bulunabilir [3].

Yazılım ürün hattı kapsamında bir ürün yapılandırması, belirli işlevsel davranışlara sahip bir ürün oluşturmak için seçilen bir küme özelliğin eşsiz kombinasyonuna karşılık gelir [6]. Belirli bir YÜH işlevselliğini etkinleştirerek veya devre dışı bırakarak farklı ürün yapılandırmalarının elde edilmesi, isteğe bağlı özelliklerin seçilmesi ile olur [4]. Yazılım ürün hatlarındaki temel zorluk, yeni bir ürün yapılandırması için, YÜH'ü temsil eden özellik modelindeki özellikleri etkinleştirme ve devre dışı bırakma sürecidir [1]. Bu süreçte, yeni ürün yapılandırmasının, özellik modelindeki ilişkiler ve kısıtlar ile uyumlu bir yapılandırma olup olmadığının denetimi yapılmalıdır [4]. Bu denetim, karmaşık ve zaman bakımından maliyetli bir işlemdir [4]. YÜH'ü temsil eden özellik modelinde bulunan isteğe bağlı özelliklerin sayısı arttıkça, ürün yapılandırması sayısı ile birlikte bu işlemin karmaşası ve zaman maliyeti özellikle yüzlerce/binlerce özelliğe sahip endüstriyel çaptaki yazılım ürün hatları için artmaktadır [4,7].

Yazılım ürün hatlarının çok sayıda farklı ürün yapılandırmasına sahip olma kapasitesi, bu ürün yapılandırmalarını tek tek üretmeyi ya da test etmeyi imkansız hale getirdiğinden yazılım ürün hatları için model-tabanlı yaklaşımlar önerilmiştir [8-10]. Bir yazılım ürün hattı oluşturmak pratikte yazılım sisteminin farklı bakış açılarını, alt sistemlerini veya davranışlarını temsil etmek için özelliklerin modellenmesini içerir [4]. Bu çalışma kapsamında, farklı boyutta YÜH'lere ait ürün yapılandırmaları, Olay Sıra Çizgesi (OSÇ) ve Özellikli Olay Sıra Çizgesi (ÖOSÇ) isimli çizgeler kullanılarak modellenmiştir.

Yazılım ürün hattı kavramıyla birlikte ortaya çıkan çok sayıda ürün yapılandırması veya bu yapılandırmaların doğrulanması vb. problemlere çözüm bulmak için literatürde farklı yaklaşımlar ortaya konmuştur. Bu çalışmada ortaya konan yaklaşımda, YÜH'e ait özellik modelinde bulunan tüm özellikleri bulunduran tam ürün (İng. Full product) yapılandırması modellenmiş ve eleme koşulları kullanılarak, tam ürünü temsil eden modelden özellik (veya özellik kümesi) eksiltilme yapılmıştır. Bu sayede, farklı ürün yapılandırmaları otomatik elde edilmiştir. Son olarak, otomatik elde edilen ürün yapılandırmalarının özellik modeline uygunluğu ve ürün yapılandırmaların ait modellerin doğruluğu kontrol edilmiştir.

Literatürde yazılım ürün hatları için model-tabanlı yaklaşımlar uygulayarak farklı ürün yapılandırmalarını elde etmeyi ve/veya test etmeyi amaçlayan çalışmalar burada özetlenmiştir. Grönniger vd. [10] tarafından ortaya konan bir çalışmada, özellik modelinde var olan bütün özellikleri içeren tam (İng. complete) model oluşturulmuş ve farklı ürün yapılandırmalarına ait modeller bu modelden parametrizasyon yoluyla elde edilmiştir. İlgili çalışmada, özellik modelindeki her bir özelliğe ait *view* ismi verilen modeller ile özellikler temsil edilmiştir [10]. Yazılım ürün hattına ait her ürün yapılandırmasının test durumlarını üretebilmek için, durum makinalarını kullanarak bir 150% model oluşturan ve bütün spesifik ürün yapılandırmaları için ayrı ayrı 100% model ismi verilen modeller elde edip, bu modellerden test durumlarını üreten çalışmalar da literatüre dahil olmuştur [11,12]. Bir başka çalışmada, yazılım ürün hattına ait ürün yapılandırmaları, tanımlı

bir çekirdek ürün ve bu üründeki değişikliklerin uygulamasını yapan bir küme delta tarafından temsil edilir [8]. İlgili çalışmada delta modelleme, durum makinaları üzerinde durum ve geçişleri eklemek veya çıkarmak için kullanılır ve bu sayede farklı ürün yapılandırmaları, sonrasında bunlara ait test durumları elde edilir [8].

Yazılım ürün hatlarının model tabanlı testinde başlıca söz edilen yaklaşımlardan biri test senaryolarının yeniden kullanılması için temel özelliklerin ve bileşenlerin yeniden kullanımından yararlanan ScenTED [13] isimli yaklaşımdır. Model-tabanlı bir diğer yaklaşım olan CADeT [14] ise, UML kullanım senaryosu ve etkinlik diyagramlarını kullanarak özellik tabanlı test takımları oluşturan bir başka önemli araştırmadır. Yazılım ürün hatlarında bir başka model-tabanlı test yöntemi olan model denetimi için, tasarımın bir sonlu makine olarak modellenmesini ve model denetimini kullanarak ürünün belirlenen özelliklere sahip olup olmadığının kontrol edilmesini sağlayan bir yaklaşım Kishi ve Noda [15] tarafından önerilmiştir. Model denetimini yazılım ürün hatlarına uygulamak için başka çeşitli yaklaşımlar önerilmiştir [16-18].

Bragança vd. [19] tarafından önerilen güncel bir çalışmada, YÜH kapsamındaki bütün ürünleri temsil eden bir model, bahsi geçen çalışma kapsamında kullanılan az-kodlu uygulama platformunun (İng. Low-code application platform) dışarı aktarım özelliği kullanılarak oluşturulmuş ve bu model, ilgili çalışma kapsamında oluşturulan alana özgü dile çevrilmiştir (İng. domain specific langugage (DSL)). Elde edilen modelden, değişkenlik alana özgü dili (İng. variability domain specific langugage (DSL)) kullanılarak, son kullanıcı tarafından seçilen özelliklere göre, farklı ürün yapılandırmalarına ait modeller elde edilmiştir [19]. Horcas vd. [20] tarafından önerilen diğer bir güncel çalışmada, YÜH'lerde değişkenliği gerçekleyecek yaklaşımlar birleştirme tabanlı (İng. composition-based), bilgi notu tabanlı (İng. annotations-based) ve karma (İng. combined) olarak sınıflandırılmış ve bilgi notu tabanlı yaklaşımları, birleştirme tabanlı yaklaşımlara entegre eden özellik modellemeye dayalı bir karma yaklaşım önerilmiştir. Buna göre ilgili çalışmada web mühendisliği alanında YÜH'lere ait web uygulamalarını oluşturmak için gerekli olan her türlü web artifaktını (sabit veya değişken) içerebilen uygulama artifakt modeli (İng. application artifact model) önerilmiştir [20]. Sabit artifaktlar tüm uygulamalar için ortaktır ve herhangi bir değişkenlik içermez [20]. Değişken artifaktlar, değişkenlik boyutuna (İng. granularity) bağlı olarak birleştirme (İng. composition) kullanılarak veya bilgi notları (İng. annotations) kullanılarak uygulanabilen değişkenliğe sahiptir. Bu çalışmada [20] değişkenlik bir özellik modeli kullanılarak gösterilmekte olup, özellik modelinden oluşturulan bir yapılandırmanın uygulama artifakt modeli ile eşlenmesi ile YÜH'e ait belirli bir web uygulaması oluşturulmaktadır.

Literatürdeki diğer çalışmalardan farklı olarak bu çalışmada kullanılan sistem modeli, kullanıcının sistemle etkileşimine odaklanan Olay Sıra Çizgeleridir. OSÇ'ler hem kullanıcı beklentilerine uygun arzu edilen (İng. desirable) sistem davranışını, hem de hatalı sistem davranışlarını modellemekte kullanılabilir [21]. Olay Sıra Çizgeleri, kullanıcı beklentilerine uygun, arzu edilen olayları, olay sıraları şeklinde modeller. Hatalı sistem davranışları ise istenmeyen (İng. undesirable) olaylar olarak, OSÇ'nin tamamlayıcı (İng. complementary) modelleri kullanılarak temsil edilebilir [22]. Bu çalışmada OSÇ'lere ek olarak, YÜH'e ait özellikleri temsil eden modelleri birbirlerinden izole şekilde bir arada bulunduran Özellikli Olay Sıra Çizgeleri sistem modeli olarak kullanılmıştır [23]. ÖOSÇ'ler özellikleri temsil eden modelleri izole olarak analiz edebilirken, ilgili özelliklere ait modelleri, tanımlı ürün yapılandırmalarına uygun olacak şekilde bir arada bulundurarak, YÜH'e bütüncül bir bakış sağlayabilir. ÖOSÇ'lerde bir özellik için değişiklik, eksiltme vb. operasyonlar olması durumunda diğer özelliklerin ve sistemin bütünün etkilenmemesini sağlayan modellerdir [23]. Bu çalışmanın, literatürdeki diğer çalışmalardan bir diğer farkı da eleme koşullarıdır. Eleme koşulları bildiğimiz kadarıyla ilk kez bu çalışmada kullanılmış olup, YÜH'e ait bütün özellikleri bir arada bulunduran tam üründen özellik eksiltilerek farklı ürün yapılandırmalarının elde edilmesini sağlar. Bu çalışmanın literatüre kazandırdığı yenilikler, tam üründe bulunan özellikler arası bağımlılıkları gösteren özellik-bağımlılık ağacı, özellikler arasındaki bağımlılıklara göre farklı özellikler arasında bulunan ortak kenarların hangi özelliğe ait olduğunu belirleyen dinamik kenar eşleme algoritması ve yeni ürün yapılandırmalarını otomatik elde etmeyi

sağlayan tam üründen özellik eksiltme algoritmasıdır.

Makalenin geri kalan bölümlerinin organizasyonu şu şekildedir: Bölüm 2'de makale kapsamında tam üründen farklı ürün yapılandırmalarını otomatik elde etmek için önerilen yaklaşım açıklanmış ve ilgili tanımlar verilmiştir. 3. Bölümde yaklaşımı sınamak için kullanılan vaka çalışmaları ve yapılan deneylere ait sonuçlar aktarılmıştır. Bölüm 4'te çalışmanın ilgili/benzer çalışmalar ile karşılaştırılmasına ve çalışmanın analizine yer verilmiştir.

2. Materyal ve Metot

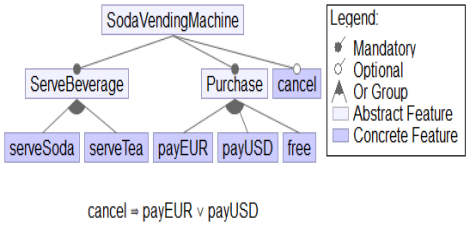
Bu bölümde çalışma kapsamında kullanılan tanımlar açıklanmış ve tam üründen farklı ürün yapılandırmalarını otomatik elde etmek için çalışma kapsamında önerilen yaklaşım anlatılmıştır.

2.1. Temel tanımlar

Bu çalışmada önerilen yöntemi açıklamak için aşağıdaki tanımlar kullanılacaktır.

Tanım 1. B = {yanlış, doğru} Boole değerlerini ile gösteren tanım kümesi ve F sonlu bir Boole değişkenleri (özellikler) kümesi olsun. Bir özellik modeli fm: (F→B)→B, F kümesi üzerinde bir önerme formülü olarak verilir [8].

Özellik modelleri Kang vd. [3] tarafından literatüre kazandırılmış olup, bu modeller, yazılım ürün hatlarında ürünlerin yapılandırma seçeneklerini ve özellikler arası bağımlılıkları göstermek için kullanılmaktadır. Şekil 1'de FeatureIDE [24] kullanılarak oluşturulan ve İçecek Otomatı YÜH'e [9] ait olan bir özellik modeli verilmiştir. Bu modelde, *SodaVendingMachine* kökü İçecek Otomatı YÜH'ü temsil etmektedir [23]. İçi dolu dairelere bağlı olan *ServeBeverage* ve *Purchase* soyut özellikleri, bu özellik modelindeki zorunlu özelliklerdir. *ServeBeverage* soyut özelliği *serveSoda* ve *serveTea* somut özelliklerini; *Purchase* soyut özelliği ise *payUSD*, *payEUR* ve *free* somut özelliklerini VEYA ilişkisi ile gruplamaktadır [23]. Soyut özellikler tarafından VEYA ilişkisi ile gruplanan *serveSoda*, *serveTea* ve *payUSD*, *payEUR* ve *free* somut özellikleri, İçecek Otomatı YÜH'e ait ürün yapılandırmalarına farklı kombinasyonlar ile katılabilir [23]. Farklı yazılım ürün hatlarında,



Şekil 1. İçecek Otomatı YÜH Özellik Modeli ([23] değiştirilerek oluşturulmuştur)

Figure 1. Soda Vending Machine SPL Feature Model (modified from [23])

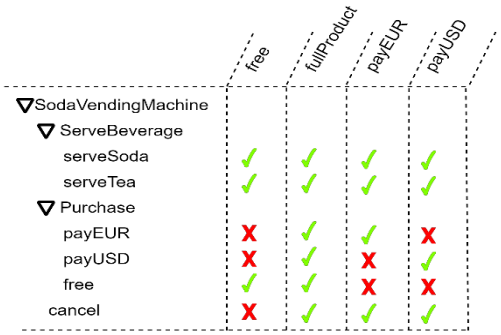
özellikler arasında, özelliklerin ürün yapılandırmasında birbirlerine alternatif olarak bulunmasını sağlayan DIŞLAYAN VEYA ilişkisi de bulunabileceği not edilmelidir [23].

İçecek Otomatı YÜH'e ait özellik modelinde içi boş daireye bağlı olan *cancel* özelliği ise, ürün yapılandırmalarında isteğe bağlı olan bir somut özelliktir [23]. İlgili özellik modelinde *cancel* özelliği ile *payEUR* veya *payUSD* özellikleri arasındaki gerektirme kısıtı görülmektedir [23]. Bu ilişki, *payEUR* veya *payUSD* özelliklerinden en az biri ürün yapılandırmasında olmadan *cancel* özelliğinin ilgili ürün yapılandırmasında olamayacağını göstermektedir [23]. Özellik modellerinde ayrıca başka bir özelliğin ürün yapılandırmasında hariç tutulduğunu belirten dışlama kısıtı da bulunabilmektedir [23].

Tanım 2. Bir ürün yapılandırması (ÜY) pc: F → B, özelliklere fm(p) = doğru eşitliğinin sağlayacak şekilde Boole değerlerinin atanmasıdır [8].

Bu çalışmada, F kümesindeki bütün özelliklere fm(p) = doğru eşitliğini sağlayacak şekilde *doğru* Boole değerinin atanmasıyla tam ürün (İng. Full product) pc_{full}: F → B elde edilir. Yani, tam ürüne ait yapılandırma ilgili yazılım ürün hattında mevcut olan ve bir arada bulunması mümkün olan bütün özelliklere sahiptir. Yazılım ürün hatlarında bir arada olması mümkün olmayan özelliklerin, DIŞLAYAN VEYA ilişkisi ya da dışlama kısıtı yüzünden aynı ürün yapılandırmasında bulunamayan özellikler olduğu not edilmelidir.

Şekil 2'de İçecek Otomatı YÜH'e ait olan farklı ürün yapılandırmaları gösterilmektedir. Örneğin, Şekil 2'deki fullProduct isimli tam ürüne ait yapılandırmada *serveSoda, serveTea, payEUR, payUSD, free* ve *cancel* özellikleri bir arada bulunmaktadır. Yazılım ürün hatlarındaki



Şekil 2. İçecek Otomatı YÜH Farklı Ürün Yapılandırmaları

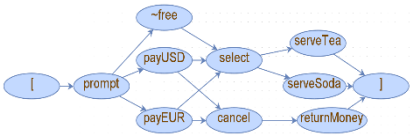
Figure 2. Soda Vending Machine SPL Different Product Configurations

birbirini dışlayıcı özellikler aynı ürün yapılandırmasında bir arada bulunamayacağı için, ilgili yazılım ürün hatlarında birden fazla tam ürün yapılandırılabileceği not edilmelidir.

Yazılım ürün hatlarında bulunan ürün yapılandırmaları için model tabanlı bir yaklaşım geliştirmek adına olay sıra çizgeleri (OSÇ) (İng. event sequence graph- ESG) ve özellikli olay sıra çizgeleri (ÖOSÇ) (İng. Featured event sequence graph- FESG) kullanılmıştır. Bu kavramlara ait tanımlar aşağıda verilmektedir.

Tanım 3. Bir olay sıra çizgesi (V, E) bir yönlü çizgedir; öyle ki V ≠ ∅ sonlu bir olaylar (düğümler ya da köşeler) kümesi ve E ⊆ V×V sonlu bir kenarlar (yaylar) kümesidir. Olay sıra çizgelerinde [ve] sözde olayları sırasıyla başlangıç ve bitiş olaylarını işaretlemek için kullanılır [21].

Şekil 3'te örnek olarak verilen OSÇ modeli, İçecek Otomatı YÜH'e ait olan *fullProduct* isimli tam ürünü temsil etmektedir. Bu ürüne ait modeldeki "*prompt*" olayı bir başlangıç olayı olup, "*serveTea*", "*serveSoda*" ve "*returnMoney*" olayları ise birer bitiş olayıdır.



Şekil 3. İçecek Otomatı YÜH *fullProduct* isimli tam ürünü temsil eden OSÇ modeli

Figure 3. ESG model of Soda Vending Machine SPL's *fullProduct*

Şekil 4. İçecek Otomatı YÜH'e ait ç-OSÇ

Figure 4. c-ESG of Soda Vending Machine SPL

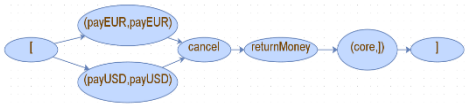
Tanım 4. Bir özellikli olay sıra çizgesi, bir çekirdek OSÇ'den (ç-OSÇ) (İng. core-ESG) ve ürün yapılandırmasına bağlı olarak bir küme özellik OSÇ'den (ö-OSÇ) (İng. feature-ESG) oluşur [23].

Yazılım ürün hattı alanında özellik, bir ürününün diğer ürünlerden ayırt edilmesini sağlayan karakteristik davranış olarak tanımlanmaktadır [25]. Bir YÜH'e ait ürün yapılandırmalarının tümünde bulunan özellik, çekirdek olarak ifade edilir[23]. Çekirdeğe ait davranış modeli, ç-OSÇ modeli ile temsil edilmektedir [23]. Bir YÜH'te bulunan ürün yapılandırmalarında farklı kombinasyonlarla bir araya getirilen ve özellik modelindeki düğümlere denk gelen özelliklere ait davranış modelleri ise ö-OSÇ'lerdir [23].

ç-OSÇ modelinde bulunan bütün düğüm ve kenarlar, YÜH'e ait bütün ürünlerin OSÇ modellerinde bulunur [23]. Şekil 4'te İçecek Otomatı YÜH'e ait ç-OSÇ modeli gösterilmektedir. Modeldeki "*prompt*" ve "*select*" olayları, ilgili YÜH'e ait çekirdek davranışı göstermektedir.

İçecek Otomatı YÜH'e ait *fullProduct* isimli tam ürün yapılandırmasındaki özelliklerden biri olan *cancel* özelliğinin ö-OSÇ modeli Şekil 5'te verilmiştir. ö-OSÇ'lerde, (OSÇ, Olay) şeklinde gösterilen ve bağlantı olayı olarak isimlendirilen düğümler bulunmaktadır [23]. ö-OSÇ modelleri bağlantı olaylarını kullanarak ç-OSÇ ve diğer ö-OSÇ'lere ait modellere bağlanır [23]. Şekil 5'te verilen ö-OSÇ modelinde görüldüğü gibi, *cancel* özelliği "*(payEUR, payEUR)*", "*(payUSD, payUSD)*" ve "*(core,])*" bağlantı olaylarını kullanarak *payEUR*, *payUSD* ve *core* modellerine bağlanabilmektedir. ç-OSÇ düğümlerinde ise bağlantı olayları bulunmamaktadır ve bu durum ç-OSÇ modellerinin yeniden kullanımını sağlamak içindir [23]. Ek olarak, sözde olaylar ve bağlantı olayları hariç her olayın ÖOSÇ ve OSÇ'lerde eşsiz olarak adlandırılacağı not edilmelidir [23].

Bu çalışmada, YÜH'te bulunan ürün yapılandırmaları hem OSÇ hem de ÖOSÇ modelleri kullanılarak temsil edilmektedir. Örneğin, İçecek Otomatı YÜH'e ait olan tam ürün



Şekil 5. *cancel* özelliğine ait ö-OSÇ modeli

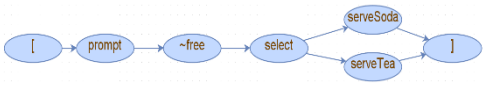
Figure 5. f-ESG model of feature *cancel*

yapılandırmasını temsil eden OSÇ modeli Şekil 3'te gösterilmektedir. İlgili YÜH'te bulunan tam ürün yapılandırmasını temsil eden ÖOSÇ modelinde ise ç-OSÇ ile birlikte *payEUR*, *payUSD*, *free*, *serveTea*, *serveSoda*, *cancel* özelliklerine ait bir küme ö-OSÇ modeli bulunmaktadır. İçecek Otomatı YÜH'e ait bütün ö-OSÇ modelleri ve ürün modelleri [26]'da verilmiştir.

Tanım 5. Eleme koşulları özelliklerin, bir ürün yapılandırmasını temsil eden OSÇ modelinden eksiltilme durumunu gösteren Boole ifadelerdir. Buna göre, bir özellik ilgili ürün yapılandırmasını temsil eden modelden eksiltilecek ise eleme koşulu *doğru* değer almaktadır. Eleme koşulları eğer *yanlış* değer alırsa ürün yapılandırmasını temsil eden modellerden eksiltilmemelidir.

Bu çalışmada, eleme koşullarının kullanım amacı, tam ürünü temsil eden OSÇ ve ÖOSÇ modellerinden özellik eksilterek farklı ürün yapılandırmalarını temsil eden OSÇ ve ÖOSÇ modellerini elde etmektir. Tam ürün yapılandırmasını temsil eden ÖOSÇ modelinden özellik eksiltmek, eleme koşullarında *yanlış* değer alan özelliklere ait ö-OSÇ modellerini, ÖOSÇ modeline ait özellikler yani ö-OSÇ'ler kümesinden eksiltmektir. Örneğin, İçecek Otomatı YÜH'teki tam ürün yapılandırmasını temsil eden ÖOSÇ modeli, *payEUR* ö-OSÇ, *payUSD* ö-OSÇ, *free* ö-OSÇ, *serveTea* ö-OSÇ, *serveSoda* ö-OSÇ, *cancel* ö-OSÇ özellik modellerine sahiptir. Tam ürün ÖOSÇ modelinden, {*payEUR* ö-OSÇ, *payUSD* ö-OSÇ, *cancel* ö-OSÇ} özellik modellerini eksilterek, {*free* ö-OSÇ, *serveTea* ö-OSÇ, *serveSoda* ö-OSÇ} özellik modelleri kümesine sahip olan *freeProduct* ürün yapılandırmasına ait ÖOSÇ modeli elde edilebilir.

Tam ürün yapılandırmasını temsil eden OSÇ modelinden özellik eksiltmek ise temelde, eksiltilecek özelliklerde bulunan bütün düğüm ve kenarları, tam ürüne ait OSÇ modelinden çıkarmaktır. Örneğin, Şekil 3'te verilen İçecek Otomatı YÜH tam ürün OSÇ modelinden, {*payEUR*, *payUSD*, *cancel*} özellikleri eksiltileceğinde, bu özelliklere ait ö-OSÇ modellerinde bulunan düğüm ve kenarlar, tam



Şekil 6. İçecek Otomatı YÜH *freeProduct* isimli ürünü temsil eden OSÇ modeli

Figure 6. ESG model of Soda Vending Machine SPL's *freeProduct*

ürüne ait OSÇ modelinden çıkarılır. Bu durumda, çıkarılacak düğümler, {"*payUSD*", "*payEUR*", "*cancel*", "*returnMoney*"} kümesi olup; çıkarılacak kenarlar da {<prompt, payUSD>, <prompt, payEUR>, <payUSD, select>, <payEUR, select>, <payUSD, cancel>, <payEUR, cancel>, <cancel, returnMoney> ve <returnMoney,]>} kümesidir. Belirlenen düğüm ve kenarlar kümesi, Şekil 3'te verilen *fullProduct* isimli tam ürüne ait OSÇ modelinden çıkarılarak, Şekil 6'da gösterilen İçecek Otomatı yazılım ürün hattına ait *freeProduct* isimli ürünün OSÇ modeli elde edilmiştir. Çıkarılacak olan düğüm ve kenarlar, çıkarılacak özelliklere ait ö-OSÇ modellerinin düğüm ve kenarları kullanılarak belirlenmektedir.

Bu çalışma kapsamında eleme koşulları, birer JSON objesi olarak temsil edilmiştir ve her bir JSON objesinin eklendiği bir JSON dosyasında tutulmuştur. Şekil 7'de İçecek Otomatı yazılım ürün hattına ait, örnek bir JSON dosyası verilmiştir. Bu dosyada, İçecek Otomatı YÜH'e ait tam ürün yapılandırmasını temsil eden ÖOSÇ ve OSÇ modelinden eksiltilecek olan {*payEUR*, *payUSD*, *cancel*} özelliklerini temsil eden JSON objeleri görülmektedir. Şekil 7'de görüldüğü üzere, *payEUR*, *payUSD* ve *cancel* özelliklerine ait eleme koşulları *yanlış* değer almıştır. Buna göre, bu özellikler tam ürünü temsil eden ÖOSÇ ve OSÇ modellerinden eksiltilecektir.

Bu çalışma kapsamında tam ürünü temsil eden modellerden, farklı ürün yapılandırmalarını temsil eden modelleri elde etmek için, eksiltilecek olan özellikler yanlış değer alan eleme koşullarına göre belirlenir. Bu doğrultuda, ilgili JSON dosyasında bulunan bütün JSON objeleri işlenir ve *yanlış* değer alan tüm eleme koşulları kullanılarak, tam üründen eksiltilecek olan özelliklerin bir listesi oluşturulur. Eksiltilecek özellikler, tam ürünü temsil eden ÖOSÇ ve OSÇ modellerinden çıkarılır.

svmProduct-eliminationCondition_payEURpayUSDcancel.json

```
{
    "eliminationConditions" :[
    {
        "ID":0,
        "conditionName":"payEUR",
        "result":true,
        "edgesToBeRemoved":[
                    "[->payEUR",
                    "prompt->payEUR",
                    "payEUR->select"
        ]
    },
    {
        "ID":1,
        "conditionName" :"payUSD",
        "result"    :true,
        "edgesToBeRemoved"  :[
                "prompt->payUSD",
                "payUSD->select"

        ]
    },
    {
        "ID":2,
        "conditionName" :"cancel",
        "result"    :true,
        "edgesToBeRemoved"  :[
                "payEUR->cancel",
                "payUSD->cancel",
                "cancel->returnMoney",
                "returnMoney->]"
        ]
    }
    ]
}
```

Şekil 7. İçecek Otomatı YÜH *payEUR*, *payUSD* ve *cancel* özelliklerini çıkaran eleme koşulu

Figure 7. Elimination condition which deducts *payEUR*, *payUSD* and, *cancel* features of Soda Vending Machine SPL

2.2. Özellik-bağımlılık ağacı

ö-OSÇ modellerinde bulunan (OSÇ, Olay) şeklinde gösterilen bağlantı olayları sebebiyle, ö-OSÇ modelleri birbirine bağlanmaktadır. ö-OSÇ modellerindeki bağlantı olayları, bazı özellikleri diğer özelliklere bağımlı hale getirmektedir. Örneğin, Şekil 5'te *cancel* özelliğine ait ö-OSÇ modelindeki "*(payEUR, payEUR)*", "*(payUSD, payUSD)*" bağlantı olayları, bu ö-OSÇ'yi "*payEUR*", "*payUSD*" ö-OSÇ'lerine bağımlı yapar. Özellikler arasındaki bağımlılıklar, tam üründen özellik eksiltilmesini etkiler, çünkü bu özellikler bir arada eksiltilmelidir. Şekil 5'te görülen "*(core,])*" bağlantı olayı, *cancel* özeliğini, *core* yani çekirdeğe bağımlı yapar ancak; çekirdek YÜH'e ait ürün yapılandırmalarının tümünde bulunan özellik olduğu için ürünlerden eksiltilemez.

İçecek Otomatı YÜH'e ait tam üründen hem *payEUR* hem *payUSD* özelliklerinin eksiltildiği bir senaryoda, *cancel* özelliği ürün yapılandırmasında var olamaz. Bunun birinci sebebi, ilgili durumda ürün davranışının etkilenmesidir. *cancel* özelliği ürüne ödemeyi iptal etme ve ödenen parayı geri alma davranışı kazandırmaktadır. *payEUR* ve *payUSD* özellikleri olmayan bir üründe, ödeme davranışı olamayacağı için, *cancel* özelliği ile ürüne dahil olan ödemeyi iptal etme davranışının da üründe olmaması gerekir. İkinci sebebi ise, *cancel* özelliğine ait ö-OSÇ modelinde bulunan bağlantı olaylarının *payEUR* ve *payUSD* ö-OSÇ modellerinde bulunan, sırasıyla "*payEUR*" ve "payUSD" olaylarına bağlantısının olmasıdır. Bu iki sebep, *cancel* özelliğinin, *payEUR* ve *payUSD* özelliklerinden en az biri olmadan tek başına ürün yapılandırmasında bulunamayan yani bağımlı bir özellik olduğunu ifade etmektedir. Sonuç olarak, *payEUR* ve *payUSD* özelliklerinin ikisi birden tam üründen eksiltildiğinde, *cancel* özelliği de bu özelliklerle birlikte eksiltilmelidir.

Tam üründe bulunan özellikler arasından, bağımlı olan ve bir arada eksiltilmesi gereken özelliklerin tespit edilmesi için, her düğümünde bir özelliğe ait OSÇ modelini bulunduran özellik-bağımlılık ağaçları bu çalışma kapsamında ortaya konmuştur.

Özellik-bağımlılık ağacı, özellikler-arası bağımlılıkları kolayca tespit ve takip edebilmek için kullanılan bir veri yapısıdır. Özellik-

Algoritma 1: Bağımlı Özellikleri Bulma

Girdi: F: tam ürünü temsil eden ÖOSÇ
Çıktı: E: ÖOSÇ'deki her bir ö-OSÇ ve bunların ebeveyn ö-OSÇ'leri (bağımlı oldukları özellikler) için eşleme veri yapısı

```
1 foreach ö-OSÇ ∈ F do
2     foreach düğüm ∈ ö-OSÇ do
3         if düğüm = (OSÇ, Olay) then
4             ebeveyn ö-OSÇ=düğümdeki OSÇ        // düğüm bağlantı olayı
5             if ö-OSÇ ≠ ç-OSÇ then
6                 (ö-OSÇ : ebeveyn ö-OSÇ) ikilisini E'ye ekle
```

bağımlılık ağacının düğümlerinde, tam ürüne ait olan bir özelliğin OSÇ modeli, düğümün ağaçtaki seviyesi ve düğümün çocuk düğümlerinin listesi tutulmaktadır. Bu ağaç yapısında her düğümün, yazılım ürün hattındaki özelliklerin bağımlılıklarına göre *N* tane çocuk düğümü bulunabilir.

Özellik-bağımlılık ağacı, genişlik yönelimli olarak oluşturulmaktadır ve her bir düğümün ağaçtaki seviyesi belirlidir. Örneğin, üç seviyeli bir ağaçta, kök düğüm ağaçta birinci ve en üst seviyededir; kök düğümün çocuk düğümleri, ağaçta ikinci seviyededir; yaprak düğümler ise ağaçta üçüncü ve en alt seviyededir.

Bir özellik-bağımlılık ağacı oluşturabilmek için tam ürüne ait ÖOSÇ modelindeki tüm ö-OSÇ'lerin bütün düğümleri dolaşılarak, (OSÇ, Olay) şeklindeki bağlantı olayları bulunur. Burada, bağlantı olayına sahip olan ö-OSÇ, bağlantı olayında bahsedilen OSÇ'ye bağımlıdır. Bu sebeple, bulunan bağlantı olayları kullanılarak, bir ö-OSÇ'nin bağımlı olduğu çekirdek hariç tüm özellikler bulunur. Bir özelliğin bağımlı olduğu özellikler çekirdek hariç bulunur çünkü, tüm özellikler ya direkt olarak çekirdeğe ya da çekirdeğe bağımlı olan diğer özelliklere bağımlıdır. Bu çalışma kapsamında, çekirdek hiçbir zaman eksiltilecek özellikler arasına eklenmez çünkü bütün özellikler, dolaylı ya da direkt olarak çekirdeğe bağımlıdır ve çekirdek, YÜH'te bulunan bütün ürünlerin temel özelliğidir. Birbirine bağımlı olan özellikleri bulmak için gerçeklenen algoritma, Algoritma 1'de verilmiştir.

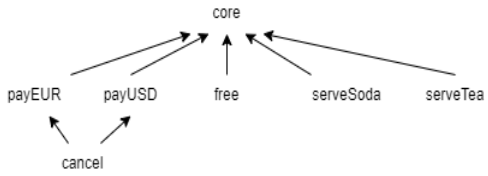
Özellik-bağımlılık ağaçları oluşturulurken, ilk olarak yazılım ürün hattına ait olan ç-OSÇ, ağacın kök düğümüne eklenir. Daha sonra, birbirine bağımlı özellikler bulunur. Özellik-bağımlılık ağaçlarında, bir özelliğin bağımlı olduğu özellikler, ilgili özelliğin bulunduğu düğümün üst ya da ebeveyn düğümlerinde bulunur. Benzer şekilde, bir düğümün alt ya da çocuk düğümleri, kendisine bağlı olan özellikleri verir. Bir özellik-bağımlılık ağacını üretmek için gerçeklenen algoritma, Algoritma 2'de verilmiştir.

Burada *payEUR*, *payUSD*, *free*, *serveSoda* ve *serveTea* özelliklerinin, ağacın kökü olan *"core"* düğümüne yani YÜH'e ait olan çekirdeğe bağımlı olduğu görülmektedir. Bu çalışmada, özellik-bağımlılık ağaçlarında çekirdeğe bağımlı olan özelliklerden daha kritik olan bilgi, iki özellik arasındaki bağımlılıktır çünkü, daha önce de bahsedildiği üzere hiçbir üründen çekirdek çıkarılmamaktadır.

Şekil 8'de verilen özellik-bağımlılık ağacı için, *cancel – payEUR* ve *cancel – payUSD* özellikleri arasındaki bağımlılık kritik bir bilgidir çünkü,

Algoritma 2: Özellik-Bağımlılık Ağacı Üretme

```
   Girdi: F: tam ürünü temsil eden ÖOSÇ
   Çıktı: A: Özellik-bağımlılık ağacı
1  D_K = ç-OSÇ ile oluşturulan kök düğüm
2  E=bağımlıÖzellikleriBulma(F)                    // Algoritma 1
3  foreach (ö-OSÇ : ebeveyn ö-OSÇ) ∈ E do
4      f = (ö-OSÇ:ebeveyn ö-OSÇ)'deki ö-OSÇ
5      D_f = f ile oluşturulan düğüm
6      S = f'nin ebeveyn ö-OSÇ'leri kümesi
7      if elemanSayısı(S) == 1 then
8          if S'deki ebeveyn ö-OSÇ == ç-OSÇ then
9              D_K'ye çocuk düğüm olarak D_f'yi ekle
10             seviye(D_f) = seviye(D_K) + 1
11         else if S'deki ebeveyn ö-OSÇ ≠ ç-OSÇ then
12             D_B = A'da seviye yönelimli arama ile ebeveyn ö-OSÇ'yi
                   içeren düğümü bul
13             D_B'ye çocuk düğüm olarak D_f'yi ekle
14             seviye(D_f) = seviye(D_B) + 1
15     else if elemanSayısı(S) >1 then
16         foreach ebeveyn ö-OSÇ ∈ S do
17             if ebeveyn ö-OSÇ ≠ ç-OSÇ then
18                 D_B = A'da seviye yönelimli arama ile ebeveyn ö-OSÇ'yi
                       içeren düğümü bul
19                 D_B'ye çocuk düğüm olarak D_f'yi ekle
20                 seviye(D_f) = seviye(D_B) + 1
```



Şekil 8. İçecek Otomatı YÜH'e ait özellik-bağımlılık ağacı

Figure 8. Feature-dependency tree of Soda Vending Machine SPL

cancel özelliğinin bir ürün yapılandırmasında bulunabilmesi için, bağımlı olduğu *payEUR* ve *payUSD* özelliklerinden en az birinin ürün yapılandırmasında bulunması gerekir. Bunlar olmadığı durumda *cancel* özelliği de tam üründen eksiltilmelidir. Ek olarak, *cancel – payEUR* ve *cancel – payUSD* özellikleri arasındaki bağımlılık *cancel* özelliğinden *payEUR* ve *payUSD* özelliklerine doğru tek yönlüdür yani, *cancel* özelliğinin ürün yapılandırmasında olmadığı durumda *payEUR* ve *payUSD* özellikleri bundan etkilenmeyecektir. Dolayısıyla, Şekil 3'te gösterilen tam üründen *payEUR* ve *payUSD* özelliklerinin ikisinin de eksiltilmesi durumunda, *cancel* özelliği de eksiltilmelidir çünkü; *cancel* özelliği *payEUR* ve *payUSD* özelliklerinden en az birisi olmadan herhangi bir ürün yapılandırmasında var olamaz.

2.3. Dinamik kenar eşleme algoritması

Bu çalışmada, tam ürünü temsil eden OSÇ modelinden, eksiltilecek özellikleri temsil eden ö-OSÇ modellerindeki kenar ve düğümler çıkarılmaktadır. Sırasıyla farklı iki ö-OSÇ'ye ait olan "*A"* ve "*B"* düğümleri arasında olan ve OSÇ modelinde bulunan kenarlara, bu çalışma kapsamında ortak kenar ismi verilmiştir. Bu kenarlar, ö-OSÇ modellerinde bulunan (OSÇ, Olay) şeklindeki bağlantı olayları sebebiyle ortak kenar olmaktadır. Şekil 3'te verilen İçecek Otomatı YÜH'e ait tam ürün OSÇ modelindeki, <*payEUR, cancel*> kenarı ortak kenara bir örnek olarak verilebilir. Bu kenardan ortak kenar olarak bahsedilmesinin nedeni *cancel* özelliğindeki "*(payEUR, payEUR)*" bağlantı olayı sebebiyle, *payEUR* özelliğinde bulunan *"payEUR"* olayı ile *cancel*'da bulunan "*cancel*" olayı arasında, tam ürüne ait OSÇ modelinde <*payEUR, cancel*> kenarı bulunmasıdır.

Tam ürünü temsil eden OSÇ modelinden, iki farklı özellik arasında ortak olan bir kenar çıkarılacağında, eksiltilecek olan özellik kümesine göre, ortak kenarın hangi özelliğe ait olacağı değişkenlik göstermektedir. Çalışma kapsamında, ortak kenarın OSÇ modelinden çıkarılıp çıkarılmamasına, ait olduğu özelliğin eksiltilecek özellikler kümesinde bulunup bulunmadığına bakılarak karar verilir. Bu kontrolü sağlamak için, bu çalışma kapsamında tam ürüne ait OSÇ modelinden çıkarılacak olan kenarların eşleneceği özellikleri tespit etmeyi amaçlayan Dinamik Kenar Eşleme Algoritması (İng. Dynamic Edge Mapping Algorithm) önerilmiş ve gerçeklenmiştir.

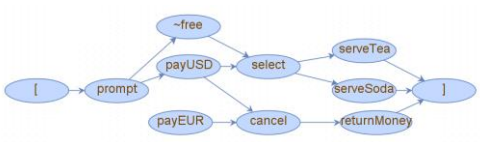
Tam üründen eksiltilecek olan özellik kümesi her değiştiğinde, bağlantı olayları sebebiyle iki farklı ö-OSÇ arasında ortak olan kenarların hangi özelliğe eşleneceği de değişir. Tam üründen özellik eksiltme sırasında, eksiltilecek özellik kümesine göre ortak kenarların eşlendiği özelliklerin değişmesine dinamik kenar eşleme adı verilmiştir. Örneğin, İçecek Otomatı yazılım ürün hattına ait tam üründen yalnızca *cancel* özelliği eksiltildiğinde, <*payEUR, cancel*> kenarı ve <*payUSD, cancel*> kenarı *cancel* özelliğine eşlenmektedir. Diğer yandan, tam üründen *payEUR* ve *payUSD* özelliklerinden ikisi birden eksiltildiğinde, ilgili kenarlar <*payEUR, cancel*> ve <*payUSD, cancel*> sırasıyla *payEUR* ve *payUSD* özelliklerine eşlenmektedir.

Ortak kenarların eksiltilen özelliklere bağlı olarak dinamik şekilde özelliklerle eşlenmesinin sebebi, statik bir eşleme yapıldığında özellik eksiltme sırasında çıkarılması gerektiği halde çıkarılamayan ortak kenarlar kalmasıdır. Tam ürüne ait OSÇ modelinden çıkarılması gerektiği halde çıkarılamayan kenarlar, yeni ürüne ait OSÇ modelinin doğru olmamasına sebep olur. Bu durum, eksiltilen özelliğe bağlı olarak ortaya çıkabilen bir sorundur. Burada statik eşleme ile bir kenarın, eksiltilen özellik kümesinden bağımsız olarak, her zaman tek bir özelliğe eşlendiği durum kastedilmektedir.

Statik bir eşleme yapıldığında özellik eksiltme sırasında ortaya çıkan problemi bir örnekle açıklamak için <*payEUR, cancel*> kenarının statik şekilde *cancel* özelliğine eşlendiği durumu düşünelim. Bu durumda, eğer tam üründen sadece *cancel* eksiltilirse, bir problem çıkmayacaktır. Ancak, tam üründen sadece *payEUR* özelliği eksiltildiğinde, *payEUR* özelliğinde bulunan tüm kenarlar tam üründen çıkarılmasına rağmen, <*payEUR, cancel*> kenarı

statik olarak cancel özelliğine eşlendiği için, tam üründen çıkarılamayacaktır.

Statik kenar eşleme yapıldığı takdirde, Şekil 3'te verilen *fullProduct* OSÇ modelinden *payEUR* özelliğinin eksiltilmesi sonucunda oluşan OSÇ modeli Şekil 9'da görülmektedir. Bu model, Şekil 3'te verilen OSÇ'den <prompt, payEUR> ve <payEUR, select> kenarları çıkarılıp, <*payEUR, cancel*> kenarı ve "*payEUR*" olayı çıkarılmayarak elde edilmiştir. Bu işlem sırasında, "*payEUR*" olayının çıkarılmamasının sebebi, <*payEUR, cancel*> kenarının kaynak düğümünün boş (İng. null) referans olmasını engellemektir. Boş referans probleminin önlenmesine rağmen, Şekil 9'da verilen OSÇ modeli yanlış (İng. invalid) bir modeldir. Bunun sebebi, "*[*" düğümünden, "*payEUR*" düğümüne bir patika (İng. path) olmamasıdır [27]. OSÇ modellerini doğrulamak için Tuğlular vd. tarafından önerilen yöntemde belirtildiği üzere bir OSÇ modelinin doğru olabilmesi için, "*[*" düğümünden, tüm düğümlere en az bir patika olması; ve tüm düğümlerden, "*]*" düğümüne en az bir patika olması gerekmektedir [27]. İlgili çalışmada [27] ek olarak, ÖOSÇ modellerini doğrulama yöntemi de önerilmiştir.



Şekil 9. Statik kenar eşleme durumunda *fullProduct* OSÇ'den *payEUR* özelliğinin eksiltilmesiyle elde edilen OSÇ

Figure 9. The ESG obtained by deducting the *payEUR* feature from the *fullProduct* ESG in case of static edge mapping

fullProduct OSÇ modelinden *payEUR* özelliğinin eksiltilmesi sırasında, <prompt, payEUR> ve <payEUR, select> kenarıyla birlikte, "*payEUR*" olayının da çıkarıldığı durumda, statik eşleme sebebiyle hala modelde olan <*payEUR, cancel*> kenarının kaynak düğümü boş (İng. null) referans haline gelecek ve ilgili OSÇ modelini kullanan algoritmaların hatalı sonuçlar döndürmesine veya çalışmamasına sebep olacaktır.

Tam üründen özellik eksiltme sırasında dinamik kenar eşleme yapabilmek için, eksiltilecek özellikler kümesinde bulunan bütün özellikler, özellik-bağımlılık ağacındaki seviyelerine göre

Algoritma 3: Dinamik Kenar Eşleme

Girdi: G: tam ürünü temsil eden OSÇ
F: tam ürünü temsil eden ÖOSÇ
E: eksiltilecek özellikler kümesi
Çıktı: K: (kenar,ö-OSÇ) ikilileri kümesi

```
1  S = özellikleriAğaçtakiSeviyelerineGöreSırala(E)
2  foreach ö − OSÇ_F ∈ F do
3      foreach kenar ∈ ö − OSÇ_F do
           /* kenar = <kaynak düğüm,hedef düğüm>
           (OSÇ,Olay) = bağlantı olayı                          */
4          if kenarın kaynak veya hedef düğümü = (OSÇ, Olay) then
5              kenar_G = G'deki kenar
6              ö-OSÇ_b = bağlantı olayındaki OSÇ
7              seviye(ö-OSÇ_F) = ö-OSÇ_F'nin S'deki sırası
8              seviye(ö-OSÇ_b) = ö-OSÇ_b'nin S'deki sırası
9              if seviye(ö-OSÇ_F) >seviye(ö-OSÇ_b) then
10                 (kenar_G:ö-OSÇ_F) ikilisini K'ye ekle
11             else if seviye(ö-OSÇ_b) >seviye(ö-OSÇ_F) then
12                 (kenar_G:ö-OSÇ_b) ikilisini K'ye ekle
13         else if kenarın kaynak veya hedef düğümü ≠ (OSÇ, Olay) then
14             kenar_G = G'deki kenar
15             (kenar_G : ö-OSÇ_F) ikilisini K'ye ekle
```

sıralanır. Bu çalışmada sıralama, özellik bağımlılık ağacının üst seviyelerdeki düğümlerinden (kök düğümden) alt seviyedeki düğümlerine (yaprak düğümlerine) olacak şekilde seçilmiştir ve her eksiltilecek özellik kümesi aynı yöntemle sıralanmaktadır. Burada, özellik-bağımlılık ağacının kök düğümünün birinci ve en üst sırada olduğu not edilmelidir. Bu sıralamanın amacı, tam ürüne ait OSÇ modelinden çıkarılacak olan kenarların eşleneceği özellikleri belirlemektir. Örneğin, İçecek Otomatı YÜH'e ait tam üründen *cancel* ve *payUSD* özelliği birlikte eksiltildiğinde, <*payUSD, cancel*> kenarı *payUSD* özelliğine eşlenmektedir çünkü *payUSD* özelliği özellik-bağımlılık ağacı üzerinde *cancel* özelliğine göre daha üst seviyededir: *payUSD* 2. seviyede, *cancel* ise 3. seviyededir (bkz. Şekil 8). Özellik-bağımlılık ağacında *cancel* özelliğine göre daha üst seviyede olan *payEUR* özelliği bu senaryoda çıkarılmadığı için, <*payEUR, cancel*> kenarı, *payUSD* ve *cancel* özelliklerinin çıkarımı sırasında, *cancel* özelliğine eşlenir.

Dinamik Kenar Eşleme Algoritması, tam ürünü betimleyen OSÇ ve ÖOSÇ modelleriyle birlikte eksiltilecek olan özellikler kümesini kullanarak (kenar, ö-OSÇ) ikililerini döndürmektedir. Dinamik kenar eşleme algoritmasına ait sözde kod, Algoritma 3'te verilmiştir.

Dinamik Kenar Eşleme Algoritmasında, ilk olarak eksiltilecek özellikler özellik-bağımlılık ağacındaki seviyelerine göre sıralanır. Algoritmanın bu adımı Algoritma 3'te 1. satırda verilmiştir. Daha sonra, ÖOSÇ'de bulunan ö-OSÇ'lerin her bir kenarı bir döngü ile gezilir (Algoritma 3 satır 2 ve 3). Her kenarın, eğer kaynak veya hedef düğümlerden biri bir bağlantı olayı ise, (OSÇ, Olay) biçiminde olan bağlantı olayı, iki aşamada çözümlenir. Birinci aşamada, bağlantı olayındaki olay ele alınır. Buna göre, eğer kaynak düğüm bağlantı olayı ise, <$Olay_{bağlantı}$, hedef> kenarı, eğer hedef düğüm bağlantı olayı ise, <kaynak, $Olay_{bağlantı}$> kenarı Algoritma 3'ün 5. satırında gösterildiği gibi tam ürüne ait OSÇ modelinde bulunur. Burada $Olay_{bağlantı}$, (OSÇ, Olay) şeklinde verilen bağlantı olayındaki "Olay"dır.

Bağlantı olayının çözümlenmesinin ikinci aşamasında, bağlantı olayındaki OSÇ'nin seviyesi, en başta verilen özelliklerin sıralamasına göre belirlenir. Döngü ile gezilen ve ÖOSÇ'ye ait olan ö-OSÇ'nin seviyesi de verilen sıralamaya göre belirlenir. İlgili seviyeler karşılaştırılarak, tam üründe bulunan kenarın kime eşleneceği belirlenir ve (kenar, ö-OSÇ) ikilisi, ikililer kümesine eklenir (Algoritma 3 satır 9-12).

Dinamik Kenar Eşleme Algoritmasında, döngü ile gezilmekte olan kenarın kaynak ya da hedef düğümlerinden ikisi de bir bağlantı olayı değil ise, ilgili kenar tam üründe bulunarak, ÖOSÇ'deki ö-OSÇ ile eşlenir (Algoritma 3 satır 13-15).

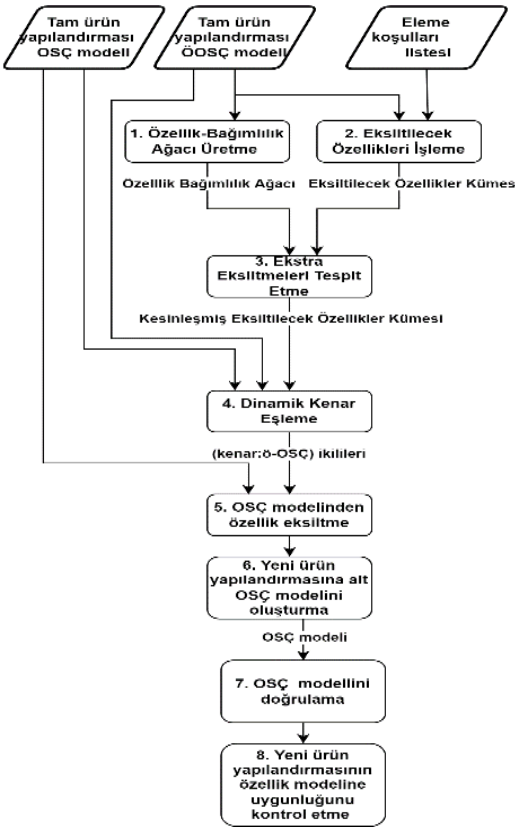
Bu çalışmada dinamik kenar eşleme algoritması kullanılarak, tam ürünü temsil eden OSÇ modelinden çıkarılacak olan kenarların ait olduğu ö-OSÇ'ler, eksiltilecek olan özellik kümesine göre belirlenir. Bu sayede, kenarların çıkarımı sonrasında oluşan OSÇ modelinde anomalilerin olması önlenmiştir. Bu anomalilere örnek olarak, "*l*" sözde düğümünden, sıradan düğümlerden birine bir yolunun (İng. path) olmaması verilebilir.

2.4. Tam üründen özellik eksiltme algoritması

Bu çalışma, tam ürün yapılandırmasını temsil eden OSÇ ve ÖOSÇ modellerinden özellik eksiltme yoluyla, farklı ürün yapılandırmalarını temsil eden OSÇ ve ÖOSÇ modellerini otomatik olarak elde etmek için eleme koşullarını kullanan bir algoritma önermektedir. Bu algoritma, ilgili YÜH'e ait tam ürün yapılandırmasını temsil eden OSÇ ve ÖOSÇ modelleriyle birlikte eksiltilecek özellikleri gösteren eleme koşullarının bir listesini kullanarak farklı bir ürün yapılandırmasını temsil eden OSÇ ve ÖOSÇ modellerini üretir. Çalışma kapsamında önerilen algoritmanın adımları Şekil 10'daki akış diyagramında gösterilmektedir.

Akış diyagramına göre, algoritmanın ilk adımında, tam ürün yapılandırmasını temsil eden ÖOSÇ modeli kullanılarak bir özellik-bağımlılık ağacı üretilmektedir. Bu çalışma kapsamında önerilen özellik-bağımlılık ağaçları, diğer özelliklere bağlantı olayları sebebiyle bağımlı olan ve tam üründen özellik eksiltme sürecini daha karmaşık hale getiren özellikleri kolayca tespit edebilmek için kullanılmıştır. Eksiltilecek özellikler, YÜH'te bulunan özellikler-arası bağımlılıkları etkilemediği için, özellik-bağımlılık ağaçları sadece ÖOSÇ modelleri kullanılarak üretilebilmektedir. Özellik-bağımlılık ağaçlarının nasıl üretileceğinin detayları Algoritma 2'de verilmiştir.

Akış diyagramında verilen algoritmanın ikinci



Şekil 10. Tam Üründen Özellik Eksiltme Algoritması akış diyagramı

Figure 10. The flow chart of deducting features from the full product algorithm

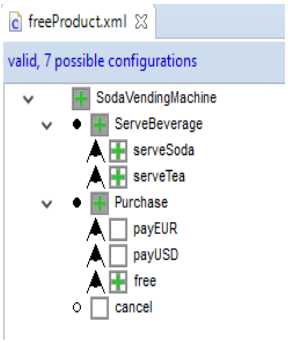
adımında, eksiltilecek olan özellikleri tespit için eleme koşullarına ait liste işlenmektedir. Daha önce de bahsedildiği gibi, bu çalışma kapsamında, her bir eleme koşulu bir JSON objesi olarak temsil edilmiş ve bir JSON dosyasına eklenmiştir. Algoritmanın bu adımında, ilgili JSON dosyası işlenmekte ve *doğru* değer alan eleme koşullarında bulunan özellikleri temsil eden ö-OSÇ'ler, tam ürüne ait ÖOSÇ modelinde tespit edilerek eksiltilecek özellikler kümesi, yani bir ö-OSÇ kümesi oluşturulmaktadır.

Tam ürünü temsil eden OSÇ modelinden özellik eksiltme ile elde edilen modelinin doğru olabilmesi için, özellik eksiltme sırasında bağımlı özellikler bulunmalı ve eksiltilecek olan özellikler kümesine eklenerek, eksiltilecek özellikler kümesi kesinleştirilmelidir. Bu sebeple akış diyagramının üçüncü adımında, eksiltilecek özellikler kümesindeki özelliklere bağımlı olan ve ekstra olarak eksiltilmesi gereken özellikler, özellik-bağımlılık ağacı kullanılarak tespit edilir ve eksiltilecek özellikler kümesine eklenerek, ilgili küme kesinleştirilir. Bu adım, Algoritma 4'te 3. satıra denk gelmektedir. *ekstraEksiltmeleriBulma* prosedürü eksiltilecek özellikleri ve özellik-bağımlılık ağacını parametre olarak alır. Bu prosedür, eksiltilecek özelliklere ait ö-OSÇ'leri özellik-bağımlılık ağacı üzerinde bularak, bunlara bağımlı olan yani çocuk/alt düğümlerindeki özellikleri temsil eden ö-OSÇ'leri de eksiltilecek özellikler arasına eklemektedir.

Akış diyagramındaki dördüncü adımda, bu çalışma kapsamında önerilen ve tam ürünü temsil eden OSÇ modelinden çıkarılacak olan ortak kenarların eşleneceği ö-OSÇ'leri belirleyen dinamik kenar eşleme algoritması kullanılmaktadır. Dinamik kenar eşleme algoritmasına ait detaylar, Algoritma 3'te verilmiştir.

Algoritma 4 girdi olarak tam ürüne ait olan OSÇ ve ÖOSÇ modellerinin birer kopyasını kabul etmektedir. Bunun sebebi, tam ürüne ait olan modellerin manipülasyona uğramamasını sağlamaktır. Akış diyagramındaki beşinci adımda, eksiltilecek özellikleri temsil eden ö-OSÇ'lerin kenarları dinamik kenar eşleme algoritması kullanılarak elde edilen (kenar:ö-OSÇ) ikililerine uygun olacak şekilde tam ürünü temsil eden OSÇ modelinin kopyasından çıkarılır (Algoritma 4, satır 5-7). Akış diyagramındaki altıncı adım ise, Algoritma 4'ün 8. satırında çağırılan prosedüre, denk gelmektedir. Bu adımda, tam ürüne ait OSÇ modelinin kopyasında kalan kenarlar bir döngü ile gezilerek, yeni ürün yapılandırmasına ait olan Ç' isimli OSÇ modeli oluşturulur. Sadece OSÇ modelinde kalan kenarlar incelendiği için, çıkarılan ö-OSÇ'lere ait olan düğümlerdeki olaylar da elenmiş olur ve yeni ürün yapılandırmasına ait OSÇ modelinde bulunmaz.

Tam üründen özellik eksiltme yöntemine dahil olan diğer bir süreç ise, tam üründen özellik eksiltme yöntemi ile elde edilen ürün yapılandırmasına ait olan OSÇ modelinin doğrulanmasıdır. Akış diyagramında yedinci adımı olan doğrulama adımı Algoritma 4'te 9. satıra denk gelmektedir. Akış diyagramındaki sekizinci ve son adımı, yeni ürün yapılandırmasının, özellik modeline uygunluğu kontrol edilmektedir. Algoritma 4'te 10. Satıra



Şekil 11. İçecek Otomatı YÜH *freeProduct* isimli ürüne ait FeatureIDE *config* dosyası

Figure 11. FeatureIDE *config* file of Soda Vending Machine SPL's *freeProduct*

denk gelen bu adımda, elde edilen yeni ürün yapılandırmasına ait FeatureIDE *config* dosyası üretilmiş ve verilen özellik modeli ile uyumlu olup olmadığı gözlenmiştir. Şekil 6'da verilen *freeProduct* isimli ürüne ait FeatureIDE *config*

dosyası Şekil 11'de gösterilmektedir. Bu çalışma kapsamında önerilen tam üründen özellik eksiltme algoritması, tam ürünü temsil eden OSÇ modelini, özellik eksiltme yoluyla işleyerek farklı ürün yapılandırmalarını temsil eden OSÇ modelleri elde etmektedir. İlgili algoritma kullanılarak yapılan deneyler ve sonuçları Bölüm 3'te ayrıntılarıyla anlatılmaktadır. Ek olarak, bu bölümde verilen algoritmalara ait kaynak kodları [28]'de verilmiştir.

3. Bulgular

Bu çalışmada, İçecek Otomatı YÜH, Banka Hesabı YÜH ve Öğrenci Yoklama Sistemi YÜH isimli üç farklı YÜH, vaka çalışması olarak kullanılmıştır.

3.1. İçecek Otomatı yazılım ürün hattı

İçecek Otomatı YÜH [29]'dan alınarak düzenlenmiştir ve en temel halinde, EUR veya USD cinsinde ödeme alarak kullanıcılarına çay veya soda servisi yapan bir otomatın davranışlarına sahiptir. İçecek Otomatı YÜH'e ait özellik modeli Şekil 1'de verilmiş olup, bu yazılım ürün hattında, tam ürün dahil toplamda 39 farklı ürün yapılandırması bulunmaktadır.

İçecek Otomatı yazılım ürün hattı, Şekil 1'de verilen özellik modelinde de görüldüğü gibi *serveSoda*, *serveTea*, *payEUR*, *payUSD*, *free* ve *cancel* isimli 6 farklı özelliğe sahiptir. Tablo 1'de, bu YÜH'e ait çekirdeği temsil eden ç-OSÇ ve özellikleri temsil eden ö-OSÇ'leri boyutları verilmiştir.

İçecek Otomatı YÜH'te birbirini dışlayan özellikler bulunmadığı için, her özelliği içeren tek bir tam ürün yapılandırması oluşturulmuştur. Tam ürünü temsil eden ve İçecek Otomatı YÜH'e ait deneylerde kullanılan OSÇ modeli Şekil 3'te verilmiştir. Bu OSÇ modeli toplamda 11 düğüm ve 15 kenara sahiptir.

İçecek Otomatı YÜH'teki özellikler arası bağımlılıkları gösteren özellik-bağımlılık ağacı, Şekil 8'de verilmiş olup, *cancel* özelliğinin

Algoritma 4: Tam Üründen Özellik Eksiltme

Girdi: G: tam ürünü temsil eden OSÇ modelinin bir kopyası
F: tam ürünü temsil eden ÖOSÇ modelinin bir kopyası
K: eleme koşullarının bir listesi
Çıktı: G': yeni ürün yapılandırmasını temsil eden OSÇ modeli
F': yeni ürün yapılandırmasını temsil eden ÖOSÇ modeli

```
1  A = özellikBağımlılıkAğacıÜretme(F)                    // Algoritma 2
2  I = eksiltilecekÖzellikleriİşleme(K, F)
3  E = ekstraEksiltmeleriBulma(I, A)
4  K = dinamikKenarEşleme(G, F, E)                        // Algoritma 3
5  foreach (kenar:ö-OSÇ) ∈ K do
6      if ö-OSÇ ∈ E then
7          G'den kenarı çıkar
8  G' = yeniÜrünYapılandırması-OSÇOluşturma(G)
9  OSÇDoğrulama(G')
10 ürünYapılandırmaDosyasıOluşturma(G')
```

Tablo 1. İçecek Otomatı YÜH ç-OSÇ ve ö-OSÇ Düğüm/Kenar Sayıları

Table 1. Soda Vending Machine SPL c-ESG and f-ESG Number of Vertices/Edges

Özellik	Düğüm Sayısı	Kenar Sayısı
core	4	1
serveSoda	4	4
serveTea	4	4
payEUR	4	4
payUSD	4	4
free	4	4
cancel	7	7

payEUR ve *payUSD* özelliklerine bağımlı olduğu görülmektedir.

Şekil 5'te de verilen İçecek Otomatı YÜH'e ait *cancel* özelliği, *payEUR* ve *payUSD* özelliklerinden en az biri bulunmadan ürün yapılandırmalarına dahil edilemez. *payEUR* ve *payUSD* özelliklerinden sadece birinin ürün yapılandırmasında bulunmasının yeterli olmasının sebebi ise *cancel* özelliğinde bulunan "*(payEUR, payEUR)*" ve "*(payUSD, payUSD)*" bağlantı olaylarına gelen kenarların kaynak düğümünün ve bu bağlantı olaylarından giden kenarların da hedef düğümünün aynı olmasıdır.

Bu çalışmada, kendilerine gelen kenarların kaynak düğümü ile, kendilerinden giden kenarların hedef düğümü aynı olan bağlantı olaylarındaki özelliklerden bir tanesi ürün yapılandırmasına eklendiğinde, bu bağlantı olaylarını içeren özellik de ürün yapılandırmasına katılabilmektedir.

3.2. Banka Hesabı yazılım ürün hattı

Banka Hesabı YÜH [30]'dan ilham alınarak düzenlenmiştir. İlgili YÜH aynı zamanda SPL2go [31] isimli web sitesinde de yayınlanmıştır. Bu YÜH'e ait her ürün, bir banka hesabını yönetmek için kullanılabilir, ancak ürünler sağladıkları bireysel özellikler bakımından farklılık gösterir. Banka Hesabı Banka Hesabı YÜH'e ait özellik modeli ve özellikler arası bağımlılıkları gösteren özellik-bağımlılık ağacı [32]'de verilmiş olup, ç-OSÇ ve ö-OSÇ modelleri [33]'te verilmiştir. Bu YÜH toplam 9 farklı özelliğe sahiptir. Bu özellikleri farklı şekillerde birleştirerek, toplamda 81 farklı ürün yapılandırması elde edilebilmektedir. Banka Hesabı YÜH'e ait özellikleri temsil eden ö-OSÇ modellerine ve çekirdeği temsil eden ç-OSÇ modeline ait düğüm ve kenar sayıları [34]'te verilmiştir.

Banka Hesabı YÜH'te *overdraft* ve *credit* özellikleri birbirini dışlayıcı olduğu için, bu özellikleri ayrı ayrı içeren *fullProduct_overdraft* ve *fullProduct_credit* isimli iki farklı tam ürün yapılandırılmıştır. *fullProduct_overdraft* tam ürünü *credit* hariç Banka Hesabı YÜH'te bulunan *deposit, withdraw, cancel deposit, cancel withdraw, overdarft, interest, interest estimation* ve *daily limit* özelliklerini içermektedir. *fullProduct_ credit* tam ürünü ise, *overdraft* hariç *credit* dahil olmak üzere bütün özellikleri içermektedir. İlgili ürünlere ait OSÇ modellerinin düğüm ve kenar sayıları [35]'te verilmiştir.

Banka Hesabı YÜH'e ait özellik bağımlılık ağacına göre, *cancelDeposit* özelliği *deposit* özelliğine, *cancelWithdraw* özelliği *withdraw* özelliğine ve *interestEstimation* özelliği *interest* özelliğine bağımlıdır. Ayrıca, *dailyLimit* özelliği *wihtdraw* ve *cancelWithdraw* özelliklerine; *overdraft* özelliği ise *cancelWithdraw* ve *dailyLimit* özelliklerine bağımlıdır.

3.3. Öğrenci Yoklama Sistemi yazılım ürün hattı

Öğrenci Yoklama Sistemi YÜH SPLOT [36] isimli web sitesinden alınmış ve düzenlenmiştir. Öğrenci Yoklama Sistemi YÜH'e ait ürünler öğrenci ve/veya öğretmenler tarafından ders ve yoklama takibi için kullanılabilir. Öğrenci Yoklama Sistemi YÜH toplamda 20 farklı özelliğe sahiptir. Bu YÜH'e ait özellik modeli, ç-OSÇ modeli ve ö-OSÇ modelleri [37]'de verilmiştir. Öğrenci Yoklama Sistemi YÜH'e ait toplam 2664 farklı ürün yapılandırması oluşturulabilmektedir. Bu YÜH'e ait çekirdeği temsil eden ç-OSÇ modelinin ve özellikleri temsil eden ö-OSÇ modellerinin düğüm ve kenar sayıları [38]'de verilmiştir.

Öğrenci Yoklama Sistemi YÜH'e ait özellik modelindeki *accessCard*, *barcode*, *fingerprint* ve *QRCode* özellikleri *submitAttendanceMethod*'a DIŞLAYAN VEYA ilişkisi ile bağlanmaktadır. *submitAttendanceMethod* zorunlu bir özellik olduğu için, bu özelliklerden sadece bir tanesi ürün yapılandırmalarında mutlaka

bulunmalıdır. Benzer şekilde, *email* ve *SMS* özellikleri de *notification* zorunlu soyut özelliğine DIŞLAYAN VEYA ilişkisi ile bağlanmıştır. Bu özelliklerden de sadece birisi ürün yapılandırmalarında mutlaka bulunmak zorundadır. Öğrenci Yoklama Sistemi YÜH'e ait bir tam ürünü oluşturmak için, *accessCard*, *barcode*, *fingerprint* ve *QRCode* özelliklerinden biri ile *email* ve *SMS* özelliklerinden biri seçilerek, geri kalan *studentAccess, teacherAccess, viewRecord, updateRecord*, *monitorAttendanceStatus*, *traceAttendanceActivity*, *viewClass*, *addNewClass*, *updateClassDetail*, *deleteClass*, *viewSchedule*, *addNewSchedule*, *editSchedule* ve *assignNewSchedule* özellikleri ile birleştirilir. *accessCard*, *barcode*, *fingerprint* ve *QRCode* özelliklerinden biri ile *email* ve *SMS* özelliklerinden birini içeren özellik kümesi 8 farklı şekilde birleştirilebileceği için, bu çalışmada Öğrenci Yoklama Sistemi YÜH'e ait 8 farklı tam ürün yapılandırması tanımlanmıştır. Öğrenci Yoklama Sistemi YÜH'te tanımlanmış 8 farklı tam ürün yapılandırmasına ait düğüm ve kenar sayıları [39]'da verilmiştir. Bu tam ürünler, *fullProduct_accessCardEmail*, *fullProduct_accessCardSMS*, *fullProduct_barcodeEmail*, *fullProduct_barcodeSMS*, *fullProduct_fingerprintEmail*, *fullProduct_fingerprintSMS*, *fullProduct_QRCodeEmail* ve *fullProduct_QRCodeSMS* olarak isimlendirilmiştir. Bu tam ürünler, *studentAccess, teacherAccess, viewRecord, updateRecord*, *monitorAttendanceStatus*, *traceAttendanceActivity*, *viewClass*, *addNewClass*, *updateClassDetail*, *deleteClass*, *viewSchedule*, *addNewSchedule*, *editSchedule* ve *assignNewSchedule* özelliklerini ve isimlerinin sonunda yazan iki özelliği içermektedir.

Öğrenci Yoklama Sistemi YÜH'te özellikler arası bağımlılıkları gösteren özellik-bağımlılık ağacı [40]'da gösterilmektedir. Buna göre, *addNewClass*, *addNewSchedule*, *deleteClass* ve *updateRecord* özellikleri *teacherAccess* özelliğine; *viewRecord* ve *monitorAttendanceStatus* özellikleri *studentAccess* özelliğine; *traceAttendanceActivity* özelliği ise *updateRecord* özelliğine bağımlıdır.

Ek olarak, *updateClassDetail* özelliği, *addNewClass* ve *teacherAccess* özelliklerine bağımlıyken; *editSchedule* özelliği de *addNewSchedule* ve *teacherAccess* özelliklerine bağımlıdır. Son olarak, *assignNewSchedule* özelliği *addNewSchedule*, *updateClassDetail* ve *editSchedule* özelliklerine bağımlıdır.

3.4. Tam üründen tek özellik eksilterek yeni ürün yapılandırması elde etme

Bu çalışma kapsamında gerçekleştirilen ilk deney kurulumunda, yazılım ürün hattına ait tam ürün veya tam ürünlerden, yazılım ürün hattına ait tüm özellikler tekli (birli) olarak eksiltilmiş ve farklı ürün yapılandırmaları elde edilmiştir. Bu deney için, Tam Üründen Özellik Eksiltme Algoritması tam ürünü temsil eden OSÇ ve ÖOSÇ modelleri ve YÜH'teki özellikler tekli şekilde kullanılarak çalıştırılmıştır. Bu deney kümesinde, tekli özellik eksiltmenin kaç milisaniye (ms) sürdüğü ve sonucunda elde edilen ürün yapılandırmalarına ait modellerin doğru olup olmadığı araştırılmıştır. Özellikler tekli olarak eksiltilirken, eksiltilen özelliklere bağımlı olan diğer özellikler de tam ürünü temsil eden modellerden çıkarılmıştır.

Çalışma kapsamındaki deneyler, Intel Core M-5Y10c 0.80 GHz işlemcili, 4 GB hafızalı, Windows 10 64-bit işletim sistemli bir dizüstü bilgisayarda gerçekleştirilmiştir. Deneyler kapsamında, tam üründen eksiltilecek bir veya bir grup özellik içeren her özelik kümesi için, tam üründen özellik eksiltme algoritması 10 kez çalıştırılarak, tam üründen özellik eksiltme süresi milisaniye bazında ölçülmüş ve bu ölçümlerin ortalaması alınarak bu makalenin sonuçlarına eklenmiştir.

İlk deney kurulumunda İçecek Otomatı YÜH için, bu YÜH'e ait *fullProduct* tam ürününden tekli şekilde eksiltilen *payEUR*, *payUSD*, *cancel*, *serveSoda*, *serveTea* ve *free* özelliklerine ait eksiltme süreleri Tablo 2'de verilmiştir. Burada, her özellik tek başına eksiltilmiş, diğer özelliklerin eksiltilmesini gerektirmemiştir. *payEUR* ve *payUSD* özelliklerine bağımlı olan *cancel* özelliği de özellikler tek tek eksiltildiği ve *payEUR* özelliği eksiltildiğinde, *payUSD* ile bağlantı kurduğu; *payUSD* özelliği eksiltildiğinde ise *payEUR* ile bağlantı kurduğu için buna dahildir. Bu deney kurulumunda Banka Hesabı YÜH için, bu YÜH kapsamında *overdraft* ve *credit* özellikleri birbirine alternatif özellikler olduğundan ayrı ayrı tanımlanan *fullProduct_overdraft* ve *fullProduct_credit* isimli iki farklı tam üründen, YÜH'te bulunan bütün özellikler tekli olarak eksiltilmiş ve eksiltme süreleri Tablo 3'e eklenmiştir.

Banka Hesabı YÜH vaka çalışması için yapılan deneyler sırasında, *fullProduct_overdraft* tam ürününde *withdraw* özelliği eksiltildiğinde *cancelWithdraw*, *dailyLimit* ve *overdraft*; *cancel-*

Tablo 2. İçecek Otomatı YÜH Tekli Özellik Eksiltme Süreleri

Table 2. Soda Vending Machine SPL Single Feature Deduction Times

Eksiltilen Özellik	Tam Üründen Özellik Eksiltme Süresi (ms)
payEUR	8.37
payUSD	8.50
cancel	6.12
serveSoda	5.92
serveTea	5.70
free	6.07

Withdraw özelliği eksiltildiğinde *dailyLimit* ve *overdraft*; *dailyLimit* özelliği eksiltildiğinde *overdraft* ve son olarak *interest* özelliği eksiltildiğinde *interestEstimation* otomatik olarak eksiltilmiştir. *fullProduct_credit* tam ürününde ise *overdraft* özelliği bulunmadığı için, *overdraft* özelliği ile ilgili ekstra otomatik eksiltmeler, özellik eksiltmelerine dahil olmamaktadır.

Tablo 3'teki sonuçlar incelendiğinde, *fullProduct_overdraft* tam ürünü için en yüksek eksiltme süresinin, *withdraw* özelliği eksiltildiğinde olduğu görülmektedir. Bunun sebebinin, *withdraw* eksiltildiğinde onunla birlikte eksiltilen *cancelWithdraw*, *dailyLimit* ve *overdraft* özellikleri olması muhtemeldir. *fullProduct_credit* tam ürününde en yüksek özellik eksiltme süresi 10.81 ms ile ise *interest* özelliği içindir. Bunu, 10.73 ms ile *cancelWithdraw* özelliği takip etmektedir. *fullProduct_credit* tam ürününde *cancelWithdraw* özelliği eksiltildiğinde *dailyLimit* özelliği, *interest* özelliği eksiltildiğinde ise *interestEstimation* özelliği eksiltilir. Başka özelliklerin eksiltilmesini gerektirmeyen *cancelDeposit*, *interestEstimation* gibi özellikleri eksiltmenin, bunu gerektiren özelliklere göre daha az zaman harcadığı Tablo 3'te göze çarpmaktadır.

İlk deney kurulumunun üçüncü adımında, Öğrenci Yoklama Sistemi YÜH kapsamında tanımlanan sekiz farklı tam üründen, bu YÜH'e ait özellikler tekli şekilde eksiltilmiştir. Bu adıma ait sonuçlar, [41]'de verilmiştir. Bu adımda, her özellik, sekiz farklı tam üründen ayrı ayrı eksiltilmiş ve bu tam ürünlerden özellikleri eksiltme sürelerinin ortalamaları bulunmuştur. Buna göre ortalama en fazla sürede eksiltilen özellik 17.04 ms ile *teacherAcces* özelliğiyken; ortalama en az sürede eksiltilen özellik 11.57 ms ile *viewSchedule* özelliğidir. *fullProduct_accessCardEmail* tam ürününden *editSchedule* özelliği eksiltilirken ölçülen 23.94 ms bu adımdaki en fazla eksiltme süresiyken, *fullProduct_barcodeEmail* tam ürününden *deleteClass* özelliği eksiltilirken ölçülen 10.85 ms bu adımdaki en az eksiltme süresidir.

Öğrenci Yoklama Sistemi YÜH'te *teacherAccess* özelliği eksiltildiğinde *addNewClass*, *addNewSchedule*, *deleteClass* ve *updateRecord* özellikleri ve bunlara bağımlı olan diğer özellikler; *studentAccess* özelliği eksiltildiğinde *viewRecord* ve *monitorAttendanceStatus* özellikleri; *updateRecord* özelliği eksiltildiğinde *traceAttendanceActivity* özelliği *addNewClass* özelliği eksiltildiğinde *updateClassDetail* özelliği ve *addNewSchedule* özelliği eksiltildiğinde de *editSchedule* özelliği eksiltilir. Ek olarak, *updateClassDetail* özelliği eksiltildiğinde *assignNewSchedule* özelliği de eksiltilir. Ancak, *assignNewSchedule* özelliği, kendilerine gelen kenarların kaynak düğümü ve kendilerinden giden kenarların hedef düğümü aynı olan "*(editSchedule,select the editable schedule)*" ve "*(addNewSchedule,add new schedule)*" bağlantı olaylarına sahiptir. Bu sebeple, *editSchedule* ve *addNewSchedule* olayla-

Tablo 3. Banka Hesabı YÜH Tekli Özellik Eksiltme Süreleri

Table 3. Bank Account SPL Single Feature Deduction Times

Eksiltilecek Özellikler	İki Farklı Tam Üründen Özellik Eksiltme Süreleri (ms)	
	overdraft	*credit*
cancelDeposit	9.24	7.82
cancelWithdraw	10.43	10.73
deposit	9.90	10.41
withdraw	10.66	10.21
interest	10.15	10.81
interestEstimation	7.20	8.03
dailyLimit	9.71	7.68
overdraft	7.34	-
credit	-	7.62

rından biri ürün yapılandırmasında olduğu sürece, tam üründen eksiltilmemesi beklenir, fakat, *editSchedule* özelliği de *addNewSchedule* özelliğine bağımlı olduğu için, *addNewSchedule* özelliği eksiltildiğinde *assignNewSchedule* ve *editSchedule* özellikleri de ürün yapılandırmasından eksiltilir. Ancak, *assignNewSchedule*, *editSchedule* özelliği eksiltildiğinde ürün yapılandırmasından eksiltilmez çünkü kendilerine gelen kenarların kaynak düğümü ve kendilerinden giden kenarların hedef düğümü aynı olan "*(editSchedule,select the editable schedule)*" ve "*(addNewSchedule,add new schedule)*" bağlantı olaylarına sahiptir.

Öğrenci Yoklama Sistemi YÜH'te bulunan tam ürünlerden {*accessCard*, *barcode*, *fingerprint*, *QRCode*} veya {email, SMS} özellik kümelerinden özellik eksiltilemeyeceği not edilmelidir çünkü bu özellik kümelerinden sadece birer özellik bir anda üründe bulunabilir ve zorunludurlar.

3.5. Tanımlı ürün yapılandırması elde etmek için tam üründen özellik eksiltme

Bu çalışma kapsamında gerçekleştirilen ikinci deney kurulumunda, vaka çalışması olarak incelenen yazılım ürün hatları için birer taban ürün (İng. base product) tanımlanmış ve bu ürüne, YÜH'te bulunan özellik kümesinden rastgele 3'er özellik ekleyerek ürün yapılandırmaları tanımlanmıştır. Bu deney kurulumunda, tanımlanan ürün yapılandırmalarına, tam ürün(ler)den özellik eksilterek ne kadar sürede ulaşılacağı araştırılmıştır.

Vaka çalışması olarak kullanılan YÜH'lere taban ürün yapılandırması oluşturmak için, özellik modelleri incelenmiş ve bu modellerin gerektirdiklerine göre özellikler seçilmiştir. Oluşturulan ürünün doğru birer ürün olması için zorunlu özellikler ve çekirdek her taban ürüne mutlaka eklenmiştir.

Bu deney kurulumu kapsamında İçecek Otomatı YÜH'te oluşturulan taban ürün için *payEUR* ve *serveSoda* özelliklerini bulunduran bir taban ürün; Banka Hesabı YÜH için *deposit* ve *withdraw* özelliklerini bulunduran bir taban ürün; Öğrenci Yoklama Sistemi YÜH için ise *teacherAccess*, *studentAccess*, *viewRecord*, *viewClass*, *viewSchedule* özellikleri ile birlikte, bu YÜH'te tanımlı olan 8 farklı tam ürüne uyumlu olacak şekilde {*accessCard*, *barcode*, *fingerprint*, *QRCode*} veya {email, SMS} özellik kümelerinden birer özellik içeren 8 farklı taban ürün yapılandırması oluşturulmuştur. İkinci deney kurulumunda vaka çalışması olan YÜH'ler için tanımlanan taban ürünlere, boyutu 3 olan birbirinden farklı özellik kümeleri eklenmiştir.

İçecek Otomatı YÜH'te, çekirdek hariç 2 özelliği olan taban ürüne 3'lü özellik kümeleri eklenerek tanımlanana ürünlerin, çekirdek hariç toplam 5 özelliği olmuştur. İçecek Otomatı YÜH'te çekirdek hariç toplam 6 özellik olduğu için, elde edilen ürün yapılandırmalarına tam üründen ulaşmak için, ilk deney kurulumundaki gibi tekli özellikler çıkarılmıştır. Banka Hesabı YÜH'te ve Öğrenci Yoklama Sistemi YÜH'te, İçecek Otomatı YÜH'e oranla daha fazla sayıda özellik bulunduğu için, tanımlı ürün yapılandırmalarına ulaşmak için tam ürünlerden özellik kümeleri eksiltilmiştir. İkinci deney kurulumunda İçecek Otomatı YÜH'e ait sonuçlar, Tablo 4'te verilmiştir.

Bu deney kurulumunda Banka Hesabı YÜH'te bulunan *fullProduct_overdraft* ve *fullProduct _credit* tam ürünlerinden, özellik eksilterek tanımlı ürün yapılandırmalarına ulaşırken alınan sonuçlar [42]'de verilmiştir. Bu kurulumda, *fullProduct_overdraft* tam ürünü için ortalama özellik eksiltme süresi 8.66 milisaniyedir. *fullProduct_overdraft* tam ürünü için en fazla sürede eksiltilen özellik kümesi 11.85 ms ile {*cancelWithdraw*, *interest*, *interestEstimation*} kümesiyken, en az sürede eksiltilen özellik kümesi 7.31 ms ile {*cancelDeposit, dailyLimit*, *overdraft*} kümesidir. Bu kurulumda, *fullProduct_credit* tam ürünü için ortalama özellik eksiltme süresi ise 7.46 milisaniyedir.

Tablo 4. İçecek Otomatı YÜH Tam Üründen Tanımlı Ürünlere Ulaşma

Table 4. Soda Vending Machine SPL Obtaining Defined Product Configurations by Deducting Feature(s) from the *fullProduct*

Taban Ürüne Eklenen Özellikler	Tam Üründen Eksiltilen Özellikler	Eksiltme Süresi (ms)
serveTea,cancel,free	payUSD	8.50
payUSD,serveTea,free	cancel	6.12
payUSD,cancel, free	serveTea	5.70
payUSD,serveTea,cancel	free	6.07

Öğrenci Yoklama Sistemi YÜH kapsamında, 8 farklı taban üründen elde edilen tanımlanan yapılandırmaları elde etmek için eksiltilen özellikler ve bu özellikleri eksiltirken geçen süreler [43]'te verilmiştir. Bu adımda, özellik kümeleri sekiz farklı tam üründen eksiltilmiş ve bu tam ürünlerden özellik kümelerini eksiltme sürelerinin ortalamaları bulunmuştur. Buna göre, {*addNewClass*, *deleteClass*, *monitorAttendanceStatus*, *updateClassDetail*, *editSchedule*, *assignNewSchedule*} özellik kümesi 12.71 ms ile ortalama en az sürede eksiltilen özellik kümesidir. {*addNewSchedule*, *deleteClass*, *updateRecord*, *editSchedule*, *traceAttendanceActivity*, *assignNewSchedule*} özellik kümesi ise 14.51 ms ile ortalama en fazla sürede eksiltilen özellik kümesidir. *fullProduct_accessCardEmail* tam ürününden {*addNewClass*, *monitorAttendanceStatus*, *updateClassDetail*, *editSchedule*, *traceAttendanceActivity*, *assignNewSchedule*} özellik kümesi eksiltilirken ölçülen 20.25 ms bu adımdaki en fazla eksiltme süresiyken, *fullProduct_barcodeEmail* tam ürününden *{addNewClass, deleteClass, updateClassDetail, editSchedule, traceAttendanceActivity, assignNewSchedule}* özellik kümesi eksiltilirken ölçülen 11.73 ms bu adımdaki en az eksiltme süresidir.

4. Tartışma ve Sonuç

Günümüzde kullanıcıların yükselen beklentileri ile birlikte artan ürün çeşitliliği, yazılım ürün hattı paradigmasını yazılım geliştiriciler için elzem hale getirmiştir. Yazılım ürün hattına ait model-tabanlı yaklaşımlarda, ürün yapılandırmalarının ve bunları temsil eden ürün modellerinin otomatik olarak elde edilmesi, zaman ve maliyet kazanımı şeklinde geri dönmektedir çünkü; yazılım ürün hattı kapsamında tanımlanan ürün yapılandırmalarının sayısı özellik sayısına ve özellikler arasındaki ilişkilere bağlıdır. Yazılım ürün hatları için otomasyon özellikle karmaşık ve geniş ölçekli sistemler için kritiktir. Bu çalışma kapsamında, yazılım ürün hattı için tanımlanan ürün yapılandırmalarını ve bunlara ait modelleri otomatik olarak elde etmeyi sağlayan, eleme koşullarını kullanarak tam üründen özellik eksiltme yöntemi önerilmiştir. Çalışmanın literatüre sunduğu yenilikler özellik-bağımlılık ağacı, dinamik-kenar eşleme algoritması ve tam üründen özellik eksiltme algoritmasıdır.

Bu çalışma kapsamında önerilen yaklaşımda [10-12]'deki yaklaşımlara benzer şekilde, yazılım ürün hattındaki bütün özellikleri bulunduran bir ürün modeli oluşturulmuştur. Bu modeller, bu çalışma dahil bahsedilen bütün çalışmalarda özellik sayısı arttıkça karmaşıklaşma riski bulundurmaktadır. [10]'da ürün yapılandırmalarını temsil eden modeller yapısal modeller iken, bu çalışma ve [11,12]'de ürün yapılandırmalarını temsil eden modeller, davranışsal modellerdir. [19]'da bu çalışmaya benzer şekilde YÜH kapsamındaki bütün ürünleri temsil eden bir model oluşturulmuş ve bu modelden farklı ürün değişkenleri elde edilmiştir ancak model, çalışma kapsamında geliştirilen alana özgü dil kullanılarak temsil edilmiştir.

Bu çalışmada önerilen yöntem üç farklı vaka çalışması üzerinde denenmiş, elde edilen sonuçlar makale kapsamında ve ilgili bağlantılarda paylaşılmıştır. Deney sonuçları farklı boyuttaki YÜH'lerden, ürün yapılandırmalarının otomatik elde edilmesinin milisaniyeler içinde gerçekleştiğini göstermektedir. Deney sonuçlarına göre, İçecek Otomatı YÜH'te bulunan 6 somut özellikten en uzun sürede eksiltilen iki özellik sırasıyla 8.50 ms ve 8.37 ms ile *payUSD* ve *payEUR* özellikleridir. *payUSD* ve *payEUR* özellikleri, eksiltme süreleri sırasıyla 5.92 ms, 5.70 ms ve 6.07 ms olan *serveSoda*, *serveTea* ve *free* özellikleri ile düğüm ve kenar sayısı bakımından birebir aynı olmasına rağmen, kendilerine bağımlı olan *cancel* özelliği sebebiyle daha fazla eksiltme sürelerine sahiptir. Banka Hesabı YÜH'te 8 somut özelliğe sahip olan *fullProduct_overdraft* tam ürünü için en uzun sürede eksiltilen özellik *withdraw* özelliğidir. *withdraw* özelliği, örneğin *dailyLimit* özelliğine göre daha az kenara sahip olmasına rağmen, kendisine bağımlı olan *overdraft* ve *cancelWithdraw* özellikleri sebebiyle daha uzun sürede eksiltilmektedir. Öğrenci Yoklama Sistemi YÜH'te bulunan 14 somut özellik arasından *teacherAccess*, sekiz farklı tam üründeki eksiltme sürelerinin ortalaması alındığında, 17.04 ms ile en uzun ortalama eksiltme süresine sahiptir. Bu YÜH'te bulunan *addNewClass* ve *addNewSchedule* özellikleri, *teacherAccess* özelliğinden düğüm ve kenar sayısı bakımından üstün olmasına rağmen, kendisine direkt olarak bağımlı olan 6 farklı özellik bulunduğu için ortalama en uzun ortalama sürede eksiltilen özellik *teacherAccess* özelliği olmuştur. *teacherAccess* özelliği, İçecek Otomatı ve Banka Hesabı YÜH'lerde bulunan özelliklerle karşılaştırıldığında da düğüm ve

kenar sayısı olarak aynı ve/veya çok yakın olduğu özellikler bulunmasına rağmen, kendisine bağımlı olan özellik sayısı diğer YÜH'lerde bulunan özelliklerden fazla olduğu için, en uzun sürede eksiltilen özelliktir.

Bu çalışmada önerilen yaklaşımın sınırlamaları YÜH'te bulunan özelliklerin sayısı arttıkça, özellik kombinasyonlarının sayısının üstel olarak artması ve bütün özellik kombinasyonlarını denemenin çok fazla kişi-saat gerektirmesidir. Çalışmada önerilen yaklaşım her vaka çalışması için iki farklı deney kümesi üzerinde denenerek doğrulanmıştır. Ek olarak, çalışma kapsamında yapılan deneylerdeki özellik eksiltmeleri, vaka çalışmalarındaki YÜH'lere ait tam ürünler üzerinde elle denenerek, önerilen yaklaşım ilgili deney kümeleri için bir kez daha doğrulanmıştır.

Çalışmada önerilen yaklaşımın otomatik şekilde ürün yapılandırması elde edilmesini sağlaması gelecek vadetmekle birlikte yaklaşımın endüstriyel çaptaki çok fazla sayıda özelliğe sahip olan YÜH'ler üzerinde denenmesi ve ölçeklenebilirliğinin gözlemlenmesi ve eleme koşullarını özellikli olay çizgeleri üzerinde kullanarak alan mühendisliği (İng. domain engineering) alanında çözümler üretmek bu çalışmanın gelecek çalışmaları arasındadır.

5. Discussion and Conclusion

Today, increasing product diversity with the rising expectations of users has made the software product line paradigm essential for software developers. In the model-based approaches of the software product line, the automatic generation of product configurations and the product models that represent them results in time and cost savings because the number of product configurations within the software product line depends on the number of features and the relationships between these features. Automation for software product lines is especially critical for complex and large-scale systems. A feature deduction method from the full product using elimination conditions, which allows for automatically obtaining the product configurations and their models defined for the software product line, is proposed within the scope of this study. The novelties this study presents are the feature-dependency tree, the dynamic-edge mapping algorithm, and the feature deduction algorithm from the full product.

Similar to the approaches in [10-12], in our proposed approach, a product model was created including all the features in the software product line. These models have the risk of becoming more complex as the number of features increases in all the studies [10-12], including ours. While the models representing product configurations in [10] are structural models, the models representing product configurations in this study and [11, 12] are behavioral models. In [19], a model representing all products within the scope of an SPL is created, similar to this study, and different product variations are obtained from this model. However, the model is represented using the domain-specific language proposed in [19].

The method proposed in this study is tested on three different case studies, and the results are presented within the scope of this study and in the given links. Experimental results show that the automatic generation of product configurations for different-sized SPLs takes place within milliseconds. According to the test results, the two features deducted in the longest time out of the six concrete features in the Soda Vending Machine SPL are *payUSD* and *payEUR* with 8.50 ms and 8.37 ms, respectively. Although the *payUSD* and *payEUR* features have the same number of vertices and edges with the *serveSoda*, *serveTea*, and *free* features, which have deduction times of 5.92 ms, 5.70 ms, and 6.07 ms respectively, they have more feature deduction times. This is because of the *cancel* feature, which depends on the *payUSD* and *payEUR* features. For the *fullProduct_overdraft* full product, which has eight concrete features in the Bank Account SPL, the feature that has been deducted in the longest time is the *withdraw* feature. Although the *withdraw* feature has fewer edges than, for example, the *dailyLimit* feature, it is deducted in a longer time due to the *overdraft* and *cancelWithdraw* features that are dependent on it. Among the 14 concrete features in the Student Attendance System SPL, *teacherAccess* has the longest average deduction time, which is 17.04 ms when the average deduction times in eight different full products is taken. Although the *addNewClass* and *addNewSchedule* features in this SPL are superior to the *teacherAccess* feature in terms of the number of vertices and the number of edges, since there are six different features that are directly dependent on it, the *teacherAccess* feature is the feature that is deducted in the

longest average time. The *teacherAccess* feature is deducted in the longest time compared to the other features in other SPLs, although there are features that are the same or very close in terms of the number of vertices and the number of edges in Soda Vending Machine and Bank Account SPLs. The reason for this is the number of dependent features it has is greater than the features in other SPLs.

The limitations of the approach proposed in this study are that the number of feature combinations increases exponentially as the number of features in the SPL increases, and it takes too many person-hours to try all feature combinations. The approach proposed in the study is validated by testing on two different sets of experiments for each case study. In addition, the feature deduction in the experiments carried out within the scope of the study is manually tested on the full products of the SPLs in the case studies, and the proposed approach was once again validated for the relevant experiment sets.

Although the proposed approach in the study is promising as it provides automatic product configuration, it is among the future work of this study to test the approach on industrial-scale SPLs with many features to observe its scalability and to generate solutions in domain engineering using elimination conditions with featured event sequence graphs.

6. Etik Kurul Onayı ve Çıkar Çatışması Beyanı

Hazırlanan makalede etik kurul izni alınmasına gerek yoktur.

Hazırlanan makalede herhangi bir kişi/kurum ile çıkar çatışması bulunmamaktadır.

Kaynakça

[1] K. Pohl, G. Böckle, F. Linden. 2005. Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, 467s.

[2] J. Whithey. 1996. Investment Analysis of Software Assets for Product Lines. Software Engineering Institute, Carnegie Mellon University,Pittsburgh, Pennsylvania. Teknik Rapor CMU/SEI-96-TR-010

[3] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson. 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute Carnegie-Mellon University Pittsburgh Pennsylvania. Teknik Rapor CMU/SEI-90-TR-021

[4] L. Machado, J. Pereira, L. Garcia, E. Figueiredo.2014. SPLConfig: Product Configuration in Software Product Line. Brazilian Conference on Software (CBSoft), Tools Session, 1-8.

[5] K. Czarnecki,U. Eisenecker. 2000. Generative Programming: Methods, Tools, and Applications. ACM Press/ Addison-Wesley Publishing Co, 864s.

[6] Siegmund, N., Ruckel, N., Siegmund, J. 2020. Dimensions of software configuration: on the configuration context in modern software development. ESEC/FSE 2020: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Kasım 2020, Sacramento, California, United States, 338-349.

[7] Benavides, D., Trinidad, P., Ruiz-Cortés, A. 2005. Automated Reasoning on Feature Models. Advanced Information Systems Engineering, 13-17 Haziran, Porto, Portekiz, 491–503.

[8] Lochau, M., Mennicke, S., Baller, H., Ribbeck, L. 2016. Incremental model checking of delta-oriented software product lines, Journal of Logical and Algebraic Methods in Programming, Cilt 85, s. 245–267. DOI: 10.1016/j.jlamp.2015.09.004.

[9] Devroey, X. ed. 2012. A Vision for Behavioural Model-Driven Validation of Software Product Lines. Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change (ISoLA 2012), 15-18 Ekim Girit, Yunanistan, 208-222.

[10] Gronniger, H., Krahn, H., Pinkernell, C., Rumpe, B. 2008. Modeling Variants of Automotive Systems using Views. Tagungsband Modellierungs-Workshop MBEFF: Modellbasierte Entwicklung von eingebetteten, Mart 2008, Frankfurt, Berlin, Almanya.

[11] Cichos, H., Oster, S., Lochau, M., Schürr, A. 2006. Model-Based Coverage-Driven Test Suite Generation for Software Product Lines. Model Driven Engineering Languages and Systems, 1-6 Ekim, Cenova, İtalya, 425–439.

[12] Weißleder, S., Lackner, H. 2013. Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines, Electronic Proceedings in Theoretical Computer Science, Cilt. 111. DOI: 10.4204/EPTCS.111.7.

[13] Reuys, A., Kamsties, E., Pohl, K., Reis, S. 2005. Model-Based System Testing of Software Product Families. Advanced Information Systems Engineering, 13-17 Haziran, Porto, Portekiz, 519–534.

[14] Olimpiew, E. M. 2008. Model-Based Testing for Software Product Lines. George Mason University, Doktora Tezi, 276s, Fairfax, Virginia, ABD.

[15] Kishi, T., Noda, N. 2006. Formal verification and software product lines. Communications of the ACM, Cilt. 49, no. 12, 73–77. DOI: 10.1145/1183236.1183270.

[16] Gruler, A., Leucker, M., Scheidemann, K. 2008. Modeling and Model Checking Software Product

Lines. Formal Methods for Open Object-Based Distributed Systems, 14-16 Haziran, Bolonya, İtalya, 113–131.

[17] Classen, A. 2011. Modelling and Model Checking Variability-Intensive Systems. University of Namur, Doktora Tezi, Namur, Belçika.

[18] Classen, A., Heymans, P., Schobbens, P.-Y., Legay, A. 2011. Symbolic model checking of software product lines. International Conference on Software Engineering, 21-28 Mayıs, Waikiki, Honolulu, HI, ABD, 321–330.

[19] Bragança, A., Azevedo, I., Bettencourt, N., Morais, C. ,Teixeira, D., Caetano, D. 2021. Towards supporting SPL engineering in low-code platforms using a DSL approach. 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, 17-22 Ekim, Şikago, ABD, 16–28.

[20] Horcas, J.-M., Cortiñas, A., Fuentes, L., Luaces, M. R. 2022. Combining multiple granularity variability in a software product line approach for web engineering. Information and Software Technology, Cilt. 148. DOI: 10.1016/j.infsof.2022.106910.

[21] Belli, F. 2001. Finite state testing and analysis of graphical user interfaces. International Symposium on Software Reliability Engineering, 27-30 Kasım, Hong Kong, 34–43.

[22] Belli, F., Budnik, C. J. 2004. Minimal Spanning Set for Coverage Testing of Interactive Systems. Theoretical Aspects of Computing, 20-24 Ekim, Guiyang, China, 220–234.

[23] Tuglular, T., Beyazit, M., Ozturk, D. 2019. Featured Event Sequence Graphs for Model-Based Incremental Testing of Software Product Lines. IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 15-19 Temmuz, Milwaukee, Wisconsin, ABD, 2019.

[24] Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G., Leich, T. 2014. FeatureIDE: An extensible framework for feature-oriented software development. Science of Computer Programming, Cilt. 79, s. 70–85, DOI: 10.1016/j.scico.2012.06.002.

[25] Anonim, Product Line Engineering Concepts: Features, Feature Models, and Feature Profiles. https://www.productlineengineering.com/concepts/features.html (Erişim Tarihi: 02.12.2022).

[26] esg4aspl. 2021. SPL-FESG-Examples İçecek Otomatı YÜH. https://github.com/esg4aspl/SPL-FESG-Examples/blob/f0ae974761009b1c40c08242921f96fde9dfe9d7/SodaVendingMachineSPL.md (Erişim Tarihi: 14.12.2022).

[27] Tuğlular, T., Beyazıt, M., Öztürk, D. 2019. Yazılım Ürün Hatları için Özellik Yönelimli Test Modellerinin Yönetimi. 13. Ulusal Yazılım Mühendisliği Sempozyumu, 23-25 Eylül, İzmir.

[28] esg4aspl. 2020. FESG-Engine Kaynak Kodları. https://github.com/esg4aspl/fesg-engine (Erişim Tarihi: 14.12.2022).

[29] Classen, A., Cordy, M., Schobbens, P.-Y., Heymans, P., Legay, A., Raskin, J.-F. 2013. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. IEEE Transactions on Software Engineering, Cilt. 39, no. 8, s. 1069–1089.DOI: 10.1109/TSE.2012.86.

[30] Thüm, T., Schaefer, I., Apel, S., Hentschel, M. 2012. Family-based deductive verification of software product lines. ACM SIGPLAN Notices, Cilt. 48, no. 3, s. 11–20. DOI: 10.1145/2480361.2371404.

[31] Thüm, T., Meinicke, J. 2013. SPL2go Banka Hesabı YÜH. http://spl2go.cs.ovgu.de/projects/54 (Erişim Tarihi: 14.12.2022).

[32] esg4aspl. 2022. esg-generation-by-feature-deduction Banka Hesabı YÜH. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/BankAccountSPL.md (Erişim Tarihi: 14.12.2022).

[33] esg4aspl. 2021. SPL-FESG-Examples Banka Hesabı YÜH. https://github.com/esg4aspl/SPL-FESG-Examples/blob/master/BankAccountSPL.md (Erişim Tarihi: 14.12.2022).

[34] esg4aspl. 2022. esg-generation-by-feature-deduction Banka Hesabı YÜH-Özellikler https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/BankAccountSPL/FeatureData.pdf (Erişim Tarihi: 14.12.2022).

[35] esg4aspl. 2022. esg-generation-by-feature-deduction Banka Hesabı YÜH-Tam Ürünler https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/BankAccountSPL/FullProductData.pdf (Erişim Tarihi: 14.12.2022).

[36] SPLOT - Software Product Line Online Tools. http://www.splot-research.org/ (Erişim Tarihi: 14.12.2022).

[37] esg4aspl. 2021. SPL-FESG-Examples Öğrenci Yoklama Sistemi YÜH. https://github.com/esg4aspl/SPL-FESG-Examples/blob/c895b0ea3d4669a13a4484c8a50757ce2a618b79/StudentAttendanceSystem.md (Erişim Tarihi: 14.12.2022).

[38] esg4aspl. 2022. esg-generation-by-feature-deduction Öğrenci Yoklama Sistemi YÜH-Özellikler. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/StudentAttendanceSystemSPL/FeatureData.pdf (Erişim Tarihi: 14.12.2022).

[39] esg4aspl. 2022. esg-generation-by-feature-deduction Öğrenci Yoklama Sistemi YÜH-Tam Ürünler. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/StudentAttendanceSystemSPL/FullProductData.pdf (Erişim Tarihi: 14.12.2022).

[40] esg4aspl. 2022. esg-generation-by-feature-deduction Öğrenci Yoklama Sistemi YÜH. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/StudentAttendanceSystemSPL.md (Erişim Tarihi: 14.12.2022).

[41] esg4aspl. 2022. esg-generation-by-feature-deduction Öğrenci Yoklama Sistemi YÜH-Tekli Özellik Eksiltme. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/StudentAttendanceSystemSPL/OneFeatureDeduction.pdf (Erişim Tarihi: 14.12.2022).

[42] esg4aspl. 2022. esg-generation-by-feature-deduction Banka Hesabı YÜH-Özellik Kümesi Eksiltme. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/BankAccountSPL/FeatureSetDeduction.pdf (Erişim Tarihi: 14.12.2022).

[43] esg4aspl. 2022. esg-generation-by-feature-deduction Öğrenci Yoklama Sistemi YÜH-Özellik Kümesi Eksiltme. https://github.com/esg4aspl/esg-generation-by-feature-deduction/blob/b11cdffebde69b0fc9fdb597f83cfaf4115211c9/StudentAttendanceSystemSPL/FeatureSetDeduction.pdf (Erişim Tarihi: 14.12.2022).