Yazılım Test Maliyet Fonksiyonlarının Otomatik Olarak Keşfedilmesi

Gülşen Demiröz ve Cemal Yılmaz

Mühendislik ve Doğa Bilimleri Fakültesi, Sabancı Üniversitesi, İstanbul, Türkiye
{gulsend, cyilmaz}@sabanciuniv.edu

Özet: Yüksek derecede özelleştirilebilir sistemlerin test edilmesi genellikle muazzam genişlikte bir konfigürasyon uzayının örneklendirilmesi ve sadece seçilen bazı konfigürasyonların test edilmesiyle gerçekleştirilir. Örneklendirme, kapsayan dizi olarak adlandırılan bir kombinatoryal objenin hesaplanması ile gerçekleştirilebilir. Geleneksel kapsayan dizilerde bütün konfigürasyonların maliyetlerinin eşit olduğu varsayılır ki bu pratik bir varsayım değildir. Test maliyetini dikkate alan kapsayan diziler hesaplamak için test maliyetinin önceden bilinmesi gerekmektedir. Test maliyeti fonksiyonunun pratik bir şekilde ifade edilebilmesi, gerek kalite kontrol sürecinin planlanabilmesi gerekse maliyeti dikkate alan kapsayan dizilerin hesaplanabilmesi açısından önem arz etmektedir. Test maliyeti fonksiyonlarının yazılım geliştirenler tarafından sağlıklı ve hatasız bir şekilde tanımlanamayacağını öngördüğümüz için maliyet fonksiyonlarını otomatik olarak keşfedecek yöntemler geliştirdik. İlk geliştirdiğimiz yöntemimizde, kapsayan bir dizideki konfigürasyonlarda sistemin verilen kalite kontrol işi için test maliyetleri ölçülerek, bu veri kümesinden çeşitli genelleştirilmiş lineer regresyon modeller oluşturulmuştur. Bu çalışmamızda maliyet fonksiyonunu hesaplamak için yeni bir yöntem daha geliştirilmiştir ve lineer regresyon modeller ile karşılaştırılmıştır. Bunun için Deney Tasarım Teorisi kullanılmıştır. Bu teorinin özellikle eleme tasarımları kısmından faydalanılmıştır. Geliştirilen yeni yöntem, verilen bir konfigürasyon uzayı ve bu uzayda yürütülmesi planlanan bir kalite güvencesi işi için eleme tasarımlarını kullanarak kalite güvencesi maliyetlerine etkisi en çok olan parametre kombinasyonlarını belirler ve bu kombinasyonları kullanarak bir maliyet modeli hesaplar. Bu modeller üç değişik kalite kontrol işleri için (1- Sistemin kodunu derleme ve yapım işi, 2- Tek bir test durumunun koşturulması işi, 3- Tüm test durumlarının koşturulması işi) iki gerçek yazılım sistemi (Apache web sunucusu ve MySQL veritabanı sunucusu) kullanılarak geliştirilmiştir. Genelleştirilmiş lineer regresyon ve eleme tasarımları ile hesaplanan maliyet modelleri istatistik bilimlerinde R-kare olarak bilinen belirleme katsayısı ölçüm metriği ile değerlendirilmiş ve maliyet hesaplamasında sırasıyla 0.92 ve 0.99 ortalama R-kare değerleriyle oldukça başarılı sonuçlar elde edilmiştir.

Anahtar kelimeler: yazılım kalite güvencesi, yazılım test maliyeti, kapsayan diziler, Deney Tasarım Teorisi, eleme tasarımları, genelleştirilmiş lineer regresyon modeli

Abstract: The testing of highly configurable systems almost always involves sampling enormous configuration spaces and testing representative instances of a system's behavior. This sampling can be done by computing a combinatorial object, called a t-way covering array (CA). The covering arrays assume that the cost of configuring the system under test is the same for all configurations, however this is not a practical assumption. To compute cost-aware covering arrays, the cost needs to be determined beforehand. Therefore, estimating the cost of a quality assurance (QA) task across a configuration space is of great importance, as the estimates can be used for planning the QA process as well as for taking cost-aware samples. However, manually creating cost models is cumbersome and error-prone, thus impractical. Therefore we have been developing automated approaches for cost model discovery in configuration spaces. In our previous work, we have computed generalized linear regression models from the data set which contains the measured costs of all configurations in a covering array for a given QA task. In this paper, we have developed another approach using Design of Experiments Theory (DoE) for automatically discovering the cost function and compared it with our previous approach based on linear regression models. Given a configuration space, a QA task of interest, and a cost of the QA task, the proposed approach first identifies important effects, i.e., combinations of option settings that affect the cost most, by using screening designs from the DoE theory, and then uses the important effects identified to fit a cost model to the observations. To evaluate the proposed approach, we used 3 different QA tasks (1- To build the system under test 2- To run a single test case 3- To run a whole test suite) on 2 different real software systems (Apache web server and MySQL database server). These models computed by both the generalized linear regression and screening designs have been evaluated by the coefficient of determination metric known as R-squared in statistics and the results have been successful with an average measure of 0.92 and 0.99.

Keywords: software quality assurance, software testing cost, Design of Experiments Theory, screening designs, generalized linear regression models, covering arrays

1 Giriş

Yüksek derecede özelleştirilebilir sistemlerin test edilmesi genellikle muazzam genişlikte bir konfigürasyon uzayının örneklendirilmesi ve sadece seçilen bazı konfigürasyonların test edilmesiyle gerçekleştirilir. Kombinatoryal etkileşim sınama yöntemleri konfigürasyon uzayını sistematik bir şekilde örneklendirip, sadece seçilen konfigürasyonları test eder. Örneklendirme, t'li kapsayan dizi olarak adlandırılan bir kombinatoryal objenin hesaplanması ile gerçekleştirilir. Bir t'li kapsayan dizi (KAD), ayrık değerler alan konfigürasyon parametreleri kümesinin her t'li altkümesi için, ilgili parametre değerlerinin her bir kombinasyonunu en az bir kere içerecek şekilde oluşturulmuş bir konfigürasyon kümesidir [1, 15, 18, 20].

Geleneksel kapsayan dizilerde bütün konfigürasyonların maliyetlerinin eşit olduğu varsayılır ki bu pratik bir varsayım değildir [4, 8]. Maliyeti dikkate alan kapsayan diziler (M-KAD) ise geleneksel KAD'lardan farklı olarak, reel test maliyetlerini göz önüne alarak kapsayan dizileri hesaplar [7, 10].

Test maliyeti fonksiyonunun pratik bir şekilde ifade edilebilmesi, gerek kalite kontrol sürecinin planlanabilmesi gerekse maliyeti dikkate alan kapsayan dizilerin hesaplanabilmesi açısından önemlidir. Örneğin; her bir parametre değerleri kombinasyonu için bir maliyet tanımlanması, her bir konfigürasyon için bir maliyet tanımlanması demektir ki konfigürasyon sayısı parametre sayısı ile üssel olarak arttığından bu pratik değildir. Dolayısı ile verilen bir konfigürasyon uzayındaki maliyet fonksiyonlarını otomatik keşfeden yöntemlere ihtiyaç vardır. Önceki çalışmamızda kapsayan diziler oluşturularak bu uzay örneklendirilmiş ve daha sonra kapsayan dizideki tüm konfigürasyonlarda gözlemlenen maliyetler kullanılarak genelleştirilmiş lineer regresyon modeller yaratılmıştır [9]. Gerçek yazılım sistemlerinde yaptığımız deneyler genelleştirilmiş lineer regresyon modellerin güvenilir maliyet modelleri keşfetmekte başarılı ve verimli olduklarını göstermiştir [9].

Bu bildirideki çalışmamızda gene aynı amaç doğrultusunda verilen bir yazılım sisteminin konfigürasyon uzayındaki maliyet fonksiyonlarını otomatik olarak keşfeden yöntemler bu kez Deney Tasarım Teorisinde yer alan eleme tasarımları [5] kullanılarak geliştirilmiştir. Ayrıca bu yeni yöntem önceki yöntemimizle [9] karşılaştırılmış ve daha başarılı olduğu gözlemlenmiştir.

Bildirinin devamında, ilk olarak literatürdeki ilgili çalışmalardan bahsedilmiştir. Daha sonra maliyet fonksiyonlarını otomatik olarak keşfeden yeni yöntem örneklerle anlatılmıştır. Bir sonraki bölümde, hesaplanan modelleri değerlendirmek üzere reel yazılım sistemleri üzerinde yapılan deneyler ve analizleri aktarılmıştır. Son bölümde ise elde edilen sonuçlar ve gelecek planları tartışılmıştır.

2 İlgili Çalışmalar

Kombinatoryal etkileşim sınama alanındaki temel bir tarama yayını [15] geleneksel kapsayan dizileri hesaplama probleminin zor bir problem, yani NP-tam (NP-complete) bir problem olduğunu söylemektedir. Ayrıca aynı yayında kapsayan diziler, girdi parametre kombinasyonlarının test edilmesi, yüksek derecede özelleştirilebilir sistemlerin test edilmesi, olay tabanlı (grafik ara yüzleri gibi) sistemlerin test edilmesi ve yazılım ürün ailelerinin test edilmesi gibi alanlarda kullanılmış olduğu anlatılmaktadır [15].

Genelleştirilmiş lineer regresyon modelleri bir çok alanda bağımlı değişkenleri modellemek için sık sık kullanılmıştır [17]. Daha da ötesi, literatürde regresyon analizi [16] başlığı altında bu alanda bir dünya yöntem de bulunmaktadır. Bizim diğer çalışmamız da karışık konfigürasyon uzaylarında maliyeti modellemek için lineer regresyon modelinin iyi bir çözüm olduğunu göstermiştir [9].

Yazılım test maliyetini modellemek için yaptığımız ilk çalışmamızda [9], bir yazılım konfigürasyon uzayı, bir kalite kontrol işi ve bu işin maliyeti için bir ölçüm verildiğinde, geleneksel kapsayan diziler oluşturularak bu uzay örneklendirilmekte ve bu seçilmiş konfigürasyonlarda test işleri çalıştırılarak her birinin maliyetleri

ölçülmektedir. Daha sonra kapsayan dizideki tüm konfigürasyonların gözlemlenen maliyetleri kullanılarak genelleştirilmiş lineer regresyon modeller yaratılmıştır. Elde edilen bu model daha önce görülmemiş konfigürasyonların maliyetini tahmin etmede kullanılmıştır. İki açık kaynak gerçek yazılım sisteminin uzaylarında yaptığımız deneyler genelleştirilmiş lineer regresyon modellerin güvenilir maliyet modelleri keşfetmekte başarılı ve verimli olduklarını göstermiştir [9].

Reel maliyet fonksiyonlarını keşfetmek için kullanılan Deney Tasarım (DoE) Teorisinde yer alan eleme tasarımları, savunma sanayiinden ilaç sanayiine hizmet sektöründen üretim sektörüne kadar birçok alanda, ürünlerin ve hizmetlerin kalitesini etkileyen başlıca faktörlerin bulunmasında ve optimize edilmesinde başarıyla kullanılmıştır [5]. Eleme tasarımlarının yazılım mühendisliği alanına uygulanabilirliği araştırılmış[6, 11]; sistem performansının modellenmesinde [2, 12] ve ileri seviyede konfigüre edilebilir sistemlerde performans regresyon testlerinin gerçekleştirilmesinde [19] kullanılmıştır.

3 Test Maliyetinin Otomatik Olarak Hesaplanması

Maliyetin güvenilir bir şekilde hızlıca otomatik olarak keşfedilmesi için geliştirilen yöntemimiz, verilen bir konfigürasyon uzayı ve bu uzayda yürütülmesi planlanan bir kalite güvencesi işi (örneğin; sistemin derlenmesi veya bir test durumunun koşturulması) için eleme tasarımlarını kullanarak test maliyetlerine etkisi en çok olan parametre kombinasyonlarını belirler ve bu kombinasyonları kullanarak bir maliyet modeli hesaplar. Bu maliyet modeli verilen bir konfigürasyonda kalite güvencesi işini yürütmenin maliyetini tahmin etmek için kullanılır.

3.1 Önerilen Yaklaşım

Maliyet modelini tahmin etmek için ilk akla gelen yöntem bahsi geçen test işini tüm konfigürasyonlarda çalıştırmak ve tüm bu maliyetleri kaydetmek olabilir. Fakat bu her bir farklı konfigürasyon için bir maliyet tanımlanması anlamına gelir ki konfigürasyon sayısı parametre sayısı ile üssel bir şekilde arttığından bu pratik değildir. Dolayısı ile bu muazzam büyüklükteki uzayı sistematik ve ekonomik bir şekilde örneklendirebilecek ve aynı zamanda da uzaydaki tüm konfigürasyonlarda yeterince doğru maliyet tahminlerinde bulunabilecek bir yönteme ihtiyaç vardır.

Bu bildiride önerilen yöntem Deney Tasarım Teorisinin (DoE) [5] eleme tasarımlarına dayanmaktadır. Eleme tasarımları ana amacı önemli düşük değerli (1-li, 2-li, veya 3-lü öyle ki k-lı etki k tane konfigürasyon parametresinin aynı andaki etkileşimi sonucunda oluşan etkidir) etkileri bulmak olan oldukça ekonomik tasarımlardır. Örneğin, Apache web sunucusunu derleme işinin maliyeti o sistem derlenirken geçen zaman olsun: sistemin `SSL` özelliği ile derlenmesi sisteme ekstra bileşenler ekleyeceğinden, `SSL` özelliği 1-li (ana etki de denebilir) bir etki olacaktır. Benzer bir şekilde, MySQL veritabanı sunucusunda bir test durumu hem `autocommit` hem de `innoDB` özellikleri varolduğunda daha uzun sürebilir çünkü `innoDB` depolama motorunun performansı `autocommit` varolduğunda yavaşlamaktadır. `innoDB` $\times$ `autocommit` birlikte 2-li etkiye güzel bir örnektir.

Bu yaklaşım, istatistikte sıklıkla kullanılan etkilerin seyrekliği prensibiyle (sparsity-of-effects principle) de uyumludur [5]. Etkilerin seyrekliği prensibi (yazılım testlerinin maliyeti konusuna uyarlandığında); test maliyetlerini, az sayıda parametre etkileşimini içeren az sayıda kombinasyonun belirlediğini, geri kalan kombinasyonların maliyete olan etkisinin ise göz ardı edilebileceğini öngörür. Bildirinin geri kalan kısmında maliyetleri belirleyen bu en önemli kombinasyonlar önemli kombinasyonlar olarak adlandırılmaktadır.

Bir konfigürasyon uzayı modeli için hesaplanmış eleme tasarımı, test maliyetine en çok etkisi olan önemli kombinasyonların (parametre değerleri kombinasyonlarının) istatistiksel açıdan güvenilir (unbiased) bir şekilde bulunmasına olanak sağlayacak şekilde seçilmiş bir konfigürasyon kümesi oluşturmaktadır.

3.2 Eleme Tasarımlarının Hesaplanması

Bu bildiride 2 değişik eleme tasarımı kullanıldı: kesirli faktöriyel (fractional factorial) ve D-optimum (D-optimal) eleme tasarımları. Ayrıca, önerilen yöntem tam (full) faktöriyel tasarımlarla karşılaştırılmak suretiyle de değerlendirildi.

Tam faktöriyel tasarımlar konfigürasyon uzaylarında yer alan olası tüm konfigürasyonları içeren tasarımlardır [5, 14]. Örneğin, ikili değeri olan n tane konfigürasyon parametresine sahip bir uzayda tam faktöriyel tasarımın boyu 2^n (ki bu olası tüm konfigürasyonların sayısıdır) olacaktır.

Kesirli faktöriyel tasarımlar ise tam faktöriyel kümesinin dikkatlice seçilmiş bir fraksiyonudur ($1/2$, $1/4$, ..., $1/2^p$ gibi) [5]. Örneğin, ikili değeri olan n tane konfigürasyon parametreli bir uzayda tam faktöriyel tasarımın boyu 2^n iken $1/2^p$ kesirine sahip bir kesirli faktöriyel tasarımın boyu $2^{(n-p)}$ olacaktır ($p < n$).

D-optimum Eleme Tasarımları ise belirli bir istatistik kriterine göre "optimum" sonucu verecek şekilde konfigürasyon uzayını örneklendiren bilgisayar destekli eleme tasarımlarıdır [5]. Bu tasarımlar meta sezgisel arama yöntemleri ile hesaplanır ve tam faktöriyel tasarımların mükemmel fraksiyonu olmak zorunda olmadıkları için genellikle kesirli faktöriyel tasarımlardan daha küçüktürler.

3.3 Önemli Etkilerin Belirlenmesi ve Maliyet Fonksiyonu

Bir eleme tasarımı yapıldıktan sonra, ki bu seçilmiş bir konfigürasyon kümesidir, belirlenen kalite güvencesi işi bu konfigürasyonlar üzerinde çalıştırılır ve reel maliyetler her bir konfigürasyon için ölçülür. Ardından, ölçülen reel maliyetlerin analizi yapılarak maliyete en çok etkisi olan önemli kombinasyonlar ve bu kombinasyonların etkileri otomatik olarak hesaplanır. DoE, maliyetlere etki eden önemli kombinasyonların etkilerini görselleştirmek için yarı-normal olasılık grafikleri gibi görsel araçlar da sunmaktadır. Önemli etkilerin yokluğunda, bu grafik y=0 yakınlarında bir doğru çizgi üzerindeki noktalardan ibarettir ve bu çizilen doğrudan ciddi şekilde uzaklaşan etkiler önemlidir.

Önemli kombinasyonlar belirlendikten sonra, bu kombinasyonlar kullanılarak önceden görülmemiş konfigürasyonların maliyetlerinin tahmin edilebilmesi için bir maliyet fonksiyonu hesaplanır. Bu fonksiyonu hesaplama yöntemi olarak ise

Tablo 1. Apache ve MySQL sunucularının konfigürasyon parametreleri.

Apache			MySQL		
no	parametre	değerleri	no	parametre	değerleri
X1	authbasic	{disable, enable}	X1	charset	{binary, armscii8}
X2	authdigest	{disable, enable}	X2	comment	{disable, enable}
X3	cacheall	{disable, enable}	X3	debug-sync	{disable, enable}
X4	cgid	{disable, enable}	X4	dependency-tracking	{disable, enable}
X5	davall	{disable, enable}	X5	embedded-server	{disable, enable}
X6	echo	{disable, enable}	X6	error-inject	{disable, enable}
X7	example	{disable, enable}	X7	gnu-ld	{disable, enable}
X8	include	{disable, enable}	X8	pthread	{disable, enable}
X9	mpm	{prefork, worker}	X9	plugins	{none, csv}
X10	proxyall	{disable, enable}	X10	shared	{disable, enable}
X11	ssl	{disable, enable}			
X12	status	{disable, enable}			

gözlemlenen gerçek değerlerle tahmin edilen değerler arasındaki farkın karesinin toplamını minimize eden standart en küçük kareler metodu kullanılmıştır [17].

3.4 Örnek Senaryo

Bu bölümde MySQL üzerinde yöntemimizi adım adım gösteren bir örnek vereceğiz:

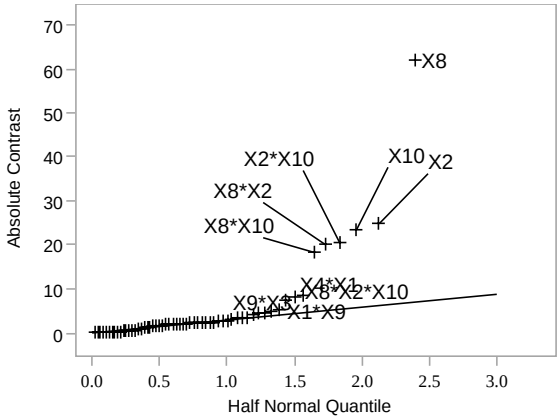
1. Adım: Bir kalite kontrol (KK) işi ve maliyetinin nasıl ölçüleceği belirlenir. Yazılım mühendisleri MySQL açık kaynak kodunda bir test durumu seçerler ve maliyeti de bu test durumunun çalıştırılma süresi olarak belirlerler.

2. Adım: Bir konfigürasyon uzayı yaratılır. Yazılım mühendisleri Tablo 1 'de verilen 10 tane konfigürasyon parametresinin tanımladığı 2^{10}=1024 boyutlu konfigürasyon uzayını oluştururlar.

3. Adım: Önemli kombinasyonların güvenli bir şekilde bulunabileceği bir eleme tasarımı hesaplanır. Önce çözünürlüğü V olan bir kesirli faktöriyel tasarımı oluşturulur [5]. Bu senaryo için oluşturduğumuz kesirli faktöriyel tasarımında 128 konfigürasyon vardır ki bu tam faktöriyel tasarımın 1/8'idir. Ama bu da mühendislere büyük gelebilirse varsayılan (default) bir D-optimum tasarım oluşturulur ki bunun boyutu sadece 60'tır (bu tam faktöriyel tasarımın sadece %6'sı ve kesirli faktöriyel tasarımın da yaklaşık olarak %47'sidir).

4. Adım: KK işi seçilen konfigürasyonlarda çalıştırılır. Yazılım mühendisleri D-optimum tasarımı seçerler. Belirlenen KK işi, tüm 60 konfigürasyonda çalıştırılır ve çalışma zamanları konfigürasyon bazında kayıt edilir.

5. Adım: Önemli kombinasyonlar tanımlanır. Deneylerin sonucunda üç tane önemli ana (1'li) kombinasyon, $X8$, $X2$ ve $X10$, ve üç tane de önemli 2'li kombinasyon, $X2{\times}X10$, $X8{\times}X2$, ve $X8{\times}X10$, belirlenir. Bu senaryoda tüm 2'li kombinasyonlarda geçen parametreler aynı zamanda önemli ana kombinasyonlarda da mevcuttur fakat bu durum her zaman böyle olmayabilir. Bu sonuçlar göstermektedir ki $X8$, $X2$ ve $X10$ parametreleri arasındaki 2'li etkileşimler bu parametrelerin tek başlarına oluşturdukları etkilerden daha fazladır.



Şekil 1. Örnek senaryo için yarı-normal olasılık grafiği (half normal probability plot).

Bu noktada yazılım mühendisleri sistem hakkındaki uzman bilgilerini kullanarak sonuçları analiz edip, önemli parametreleri biraz daha kontrol etmek isterlerse, Bölüm 3.3'de bahsi geçen yarı-normal olasılık grafiği [3] gibi analiz teknikleri kullanabilirler. Örneğin, Şekil 1'de bizim bu örnek senaryo için oluşturduğumuz yarı-normal olasılık grafiği de otomatik analizin sonuçlarını doğrular niteliktedir: 6 tane önemli kombinasyon tanımlanmıştır.

6. Adım: Gözlemlenen değerlere uyan bir maliyet modeli hesaplanır. Önemli kombinasyonlar belirlendikten sonra sadece bu önemli etkilerden oluşan, eleme tasarımlarından oluşturulmuş gözlemlere göre hesaplanan bir maliyet modeli hesaplanır. Şekil 2 bu örnek senaryo için hesaplanan maliyet fonksiyonunu göstermektedir. Bu modelin kesen değeri 59.40 olup, ayrıca model her bir önemli parametre değerleri kombinasyonu için de bir katsayı içermektedir. Katsayı pozitif ise o kombinasyon maliyeti arttırmaktadır, katsayı negatif ise de maliyeti azaltmaktadır. Örneğin, bir konfigürasyonda $X8$=1 ise, tahmin edilen maliyet 63 birim arttırılmaktadır. Aksi takdirde, 63 birim azaltılmaktadır.

Şekil 2'deki modelin oluşturulduğu D-optimum tasarım veri kümesi üzerinde test edildiğinde modelin R-kare (Bölüm 4.2) değeri 0.938 olmaktadır.

7. Adım: Hesaplanan model maliyeti tahmin etmek için kullanılır. Daha önceden karşılaşılmamış yeni bir konfigürasyon verildiğinde, hesaplanan model bu test durumunu koşturmanın maliyetini tahmin etmek için kullanılır. Örneğin, parametre değerleri, $X8$=1, $X2$=0, ve $X10$=1 (diğer parametre değerleri ne olursa olsun) olan bir konfigürasyonun tahmin edilen maliyeti $59.4 + 63 + 20.86 - 20.54 + 21.08 + 21.37 - 20.03$=145.14 birim olacaktır. Bu toplamdaki terimler sırasıyla: kesen, 1-li kombinasyonların ($X8$, $X2$, ve $X10$) ve 2-li kombinasyonların ($X2{\times}X10$, $X8{\times}X2$, ve $X8{\times}X10$) katsayılarıdır.

$$
\begin{aligned}
\mathbf{maliyet}(\mathbf{konfig}) = 59.4 + \\
\left\{ \begin{array}{c} konfig.X8 = 0 \Rightarrow -63 \\ konfig.X8 = 1 \Rightarrow 63 \end{array} \right\} + \\
\left\{ \begin{array}{c} konfig.X2 = 0 \Rightarrow 20.86 \\ konfig.X2 = 1 \Rightarrow -20.86 \end{array} \right\} + \left\{ \begin{array}{c} konfig.X10 = 0 \Rightarrow 20.54 \\ konfig.X10 = 1 \Rightarrow -20.54 \end{array} \right\} + \\
\left\{ \begin{array}{l} konfig.X2 = 0 \Rightarrow \left\{ \begin{array}{l} konfig.X10 = 0 \Rightarrow -21.08 \\ konfig.X10 = 1 \Rightarrow \ 21.08 \end{array} \right. \\ konfig.X2 = 1 \Rightarrow \left\{ \begin{array}{l} konfig.X10 = 0 \Rightarrow \ 21.08 \\ konfig.X10 = 1 \Rightarrow -21.08 \end{array} \right. \end{array} \right\} + \\
\left\{ \begin{array}{l} konfig.X8 = 0 \Rightarrow \left\{ \begin{array}{l} konfig.X2 = 0 \Rightarrow -21.37 \\ konfig.X2 = 1 \Rightarrow \ 21.37 \end{array} \right. \\ konfig.X8 = 1 \Rightarrow \left\{ \begin{array}{l} konfig.X2 = 0 \Rightarrow \ 21.37 \\ konfig.X2 = 1 \Rightarrow -21.37 \end{array} \right. \end{array} \right\} + \\
\left\{ \begin{array}{l} konfig.X8 = 0 \Rightarrow \left\{ \begin{array}{l} konfig.X10 = 0 \Rightarrow -20.03 \\ konfig.X10 = 1 \Rightarrow \ 20.03 \end{array} \right. \\ konfig.X8 = 1 \Rightarrow \left\{ \begin{array}{l} konfig.X10 = 0 \Rightarrow \ 20.03 \\ konfig.X10 = 1 \Rightarrow -20.03 \end{array} \right. \end{array} \right\}
\end{aligned}
$$

Şekil 2. Sadece önemli etkiler kullanılarak hesaplanan maliyet modeli.

4 Deneysel Çalışmalar ve Sonuçlarının Analizi

Önerilen yöntemi değerlendirmek için bir dizi deneylerde hesaplanan maliyet modellerinin gerçek maliyetleri tahmin etmedeki başarıları karşılaştırıldı.

Önerilen yöntem 3 değişik kalite güvencesi işi kullanılarak değerlendirildi:

1. **KK1**: Sistemin kodunu derleme ve yapım işi; özellikle sürekli entegrasyon senaryolarında oldukça önemlidir.
2. **KK2**: Tek bir test durumunun koşturulması işi; özellikle regresyon senaryolarında oldukça önemlidir. Deneylerde Apache sunucusu için 242 adet ve MySQL sunucusu için 826 adet bu sistemleri geliştirilenler tarafından yazılmış açık kaynak test durumu kullanılmıştır.
3. **KK3**: Tüm test durumlarının koşturulması işi; özellikle günlük sistem yapımı senaryolarında oldukça önemlidir.

Bu çalışmanın amacı; belirlenen önemli parametre kombinasyonları kullanılarak hesaplanan maliyet modellerinin gerçek maliyetleri tahmin etmedeki başarısının değerlendirilmesiydi. Bu amaç için; eleme tasarımları kullanılarak maliyet modelleri hesaplandıktan sonra bu modeller konfigürasyon uzayındaki bütün konfigürasyonların maliyetlerinin tahmin edilmesi için kullanıldı.

Çalışmada dikkat edilen diğer bir husus ise maliyeti modelleyen fonksiyonların terim sayılarının tahminlerin doğruluğunu etkilemeden azaltılmasıydı. Bu sebeple her bir tasarımdan tüm 1-li ve 2-li etkileri kullanan (*TümEtkiler*) ve sadece eleme tasarımı sonucunda bulunan önemli 1-li ve 2-li etkileri kullanan

(*ÖnemliEtkiler*) 2 tür model oluşturuldu. TümEtkiler modellerinin terim sayısı 12 parametreli uzayda Apache için 79 ve 10 parametreli uzayda MySQL için 56 iken, ÖnemliEtkiler modellerinin ortalama terim sayısı Apache için 3.58 ve MySQL için 3.62 oldu. Eğer her iki model türü de benzer tahminler üretirse tabii ki de çok daha az terime sahip ÖnemliEtkiler modelleri tercih edilecektir.

4.1 Çalışma Kurulumu

Deneylerde yüksek derecede özelleştirilebilen gerçek yazılım sistemleri olan Apache v2.2 ve MySQL v5.1 sunucuları kullanıldı. Bu yazılım sistemlerindeki çok sayıda konfigürasyon parametrelerinden seçtiklerimiz Tablo 1'de verilmiştir. Bu çalışmada nispeten az sayıda (10 ve 12) konfigürasyon parametresi kullanılmasının nedeni eleme tasarımlarını aynı uzaydaki tam faktöriyel tasarımlarıyla karşılaştırılması içindir. Deneylerde kullanılan Tablo 1'de 12 ve 10 tane konfigürasyon parametresi, sırasıyla 2^{12} ve 2^{10} konfigürasyondan oluşan uzayları tanımlarlar. Eleme tasarımlarını hesaplamak için JMP istatistik yazılım paketi kullanıldı [13].

4.2 Değerlendirme Kriterleri

Ölçüm metrikleri olarak istatistik bilimlerinde R-kare (R^2) olarak bilinen belirleme katsayısı (coefficient of determination) ve CV(RMSE) olarak bilinen kök ortalama kare hatasının (Root Mean Square Error: RMSE) varyasyon katsayısı (coefficient of variation: CV) kullanıldı [14, 17].

$$R^2 = 1 - \frac{\sum_i (c_i - \hat{c}_i)^2}{\sum_i (c_i - \bar{c})^2}, \tag{1}$$

R^2 ne kadar 1'e yakınsa o model o kadar iyidir.

$$CV(RMSE) = \frac{RMSE}{\bar{c}}, \tag{2}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{c}_i - c_i)^2}{n}}. \tag{3}$$

CV(RMSE) ne kadar küçükse o model o kadar iyidir. Yukarıdaki c_i ve $\hat{c}_i$, i'ninci konfigürasyonun verilen bir KK işi için sırasıyla gözlemlenen ve tahmin edilen maliyetleridir. $\bar{c}$ ise gözlemlenen maliyetlerin ortalamasıdır.

4.3 Çalışmanın Sonuçları ve Analizi

Tablo 2'de deneylerin sonuçları her iki yazılım sistemi için de verilmiştir. Tablodaki ilk kolon deneysel tasarımları göstermektedir: *TamFakt* - Tam Faktoriyel, *KesirFakt* - Kesirli Faktoriyel, *DOptKF* - kesirli faktoriyel tasarımı ile aynı boyuttaki D-optimum, *DOptKF* - varsayılan değer boyutundaki D-optimum eleme tasarımlarıdır; *Lineer-KADt4* - 4'lü ve *Lineer-KADt3* - 3'lü kapsayan dizilerden hesaplanan lineer regresyon modelleridir.

Tablo 2. Eleme tasarımlarıyla hesaplanan modellerin kapsayan diziler kullanılarak hesaplanan lineer regresyon modelleriyle ve birbirleriyle karşılaştırılması.

		Apache				MySQL			
Deneysel	Test	Tüm Etkiler		Önemli Etkiler		Tüm Etkiler		Önemli Etkiler	
Tasarım	işi	R^2	CV	R^2	CV	R^2	CV	R^2	CV
TamFakt	KK 1	0.9938	0.0055	0.9907	0.0068	0.9931	0.0115	0.9928	0.0118
KesirFakt	KK 1	0.9922	0.0062	0.9882	0.0076	0.9901	0.0138	0.9927	0.0119
DOptKF	KK 1	0.9917	0.0064	0.9856	0.0084	0.9897	0.0141	0.9927	0.0119
DOptVAR	KK 1	0.9727	0.0116	0.9902	0.0070	0.9820	0.0187	0.9927	0.0119
Lineer-KADt4	KK 1	0.8987	0.0209	0.7628	0.0309	0.8508	0.0728	0.7426	0.0913
Lineer-KADt3	KK 1	0.9500	0.0155	0.7000	0.0373	0.9851	0.0168	0.9822	0.0182
TamFakt	KK 2	0.9992	0.0200	0.9991	0.0208	0.9785	0.1190	0.9773	0.1227
KesirFakt	KK 2	0.9989	0.0232	0.9765	0.0681	0.9700	0.1397	0.9771	0.1234
DOptKF	KK 2	0.9989	0.0230	0.9991	0.0209	0.9719	0.1376	0.9779	0.1220
DOptVAR	KK 2	0.9977	0.0342	0.9991	0.0210	0.9610	0.1702	0.9769	0.1245
Lineer-KADt4	KK 2	0.9916	0.0535	0.9953	0.0250	0.8507	0.4472	0.7624	0.3631
Lineer-KADt3	KK 2	0.9981	0.0304	0.9986	0.0252	0.9534	0.1693	0.9589	0.1481
TamFakt	KK 3	0.9998	0.0093	0.9986	0.0226	0.9972	0.0267	0.9964	0.0306
KesirFakt	KK 3	0.9997	0.0110	0.9986	0.0226	0.9955	0.0341	0.9962	0.0312
DOptKF	KK 3	0.9997	0.0106	0.9986	0.0226	0.9964	0.0304	0.9963	0.0307
DOptVAR	KK 3	0.9991	0.0178	0.9985	0.0227	0.9939	0.0397	0.9965	0.0302
Lineer-KADt4	KK 3	0.9983	0.0234	0.9988	0.0202	0.9749	0.0776	0.8687	0.1174
Lineer-KADt3	KK 3	0.9993	0.0153	0.9990	0.0188	0.9936	0.0401	0.9955	0.0338

İlk gözlemlenen sonuç eleme tasarımlarının kapsayan dizilerle hesaplanan genelleştirilmiş lineer regresyon modellerinden daha başarılı olduğudur. Tüm eleme tasarımlarının (tam faktöriyel hariç) ortalama R^2 ve $CV(RMSE)$ değerleri 0.9907 ve 0.0385 iken, tüm kapsayan dizilerle hesaplanan lineer regresyon modellerinin ortalama değerleri 0.9254 ve 0.0797 oldu. Bu sonuç eleme tasarımlarının hesapladığı modellerin kapsayan dizilerle hesaplanan lineer regresyon modellerden daha iyi olduğunu söylemektedir. Bunun yanında eleme tasarımlarının (tam faktöriyel hariç) boyutları kapsayan dizilerden ortalama olarak 4.94 kat daha çoktur (konfigürasyon sayısı ortalama 30.75'ten 152'ye çıkmıştır)(bkz. Tablo 3).

Diğer bir gözlem tam faktöriyel tasarımlardan çok daha küçük boylardaki eleme tasarımlarının tam faktöriyel tasarımlarla eşdeğer maliyet modelleri üretebildikleri oldu. Tüm eleme tasarımlarının ortalama R^2 ve $CV(RMSE)$ değerleri 0.9907 ve 0.0385 iken tam faktöriyel tasarımlarının 0.9930 ve 0.0339 oldu. Bu sonuç daha küçük boylardaki eleme tasarımlarının hesapladığı modellerin tam faktoriyel tasarımlarının hesapladığı modeller kadar iyi olduğunu söylemektedir.

Diğer bir sonuç farklı eleme tasarımlarının da birbirlerine yakın değerler elde ettiğidir. Kesirli faktöriyel (KesirFakt) tasarımlarından elde edilen modellerin ortalama R^2 ve CV $(RMSE)$ değerleri 0.9896 ve 0.0411; DOptKF modellerinin ortalama R^2 ve CV $(RMSE)$ değerleri 0.9915 ve 0.0366; ve DOptVAR modellerinin ortalama R^2 ve $CV(RMSE)$ değerleri 0.9884 ve 0.0425 oldu. Üstelik DOptVAR tasarımları bunu, DOptKF ve KesirFakt modellerine göre, Apache için %67 ve MySQL için %53 daha az konfigürasyon kullanarak başardı (Tablo 3).

Tablo 3. Tasarımların boyutları ve tam kapsamlı tasarıma göre azaltılma yüzdeleri.

deneysel tasarım	Apache boyut	azaltılma	MySQL boyut	azaltılma
TamFakt	4096	% 0	1024	%0
KesirFakt	256	%93.8	128	%87.5
DOptKF	256	%93.8	128	%87.5
DOptVAR	84	%97.9	60	%94
Lineer-KADt4	44	%98.9	40	%96
Lineer-KADt3	21	%99.4	18	%98

Ayrıca eleme tasarımlarının TümEtkiler ve ÖnemliEtkiler modelleri birbirleriyle karşılaştırıldığında ise; ÖnemliEtkiler modellerinin ortalama olarak R^2 ve $CV(RMSE)$ değerleri 0.9912 ve 0.0381 iken TümEtkiler modellerinin ortalama R^2 ve $CV(RMSE)$ değerlerinin 0.9901 ve 0.0389 olduğu gözlemlendi. Ama ÖnemliEtkiler modellerinin terim sayısı TümEtkiler modellerindeki terim sayısından %95 daha azdı (terim sayısı ortalama 67.5'tan 3.73'e düşmüştür). Bu %95 oranında azaltılan terim sayısına karşın ortalama R^2 ve $CV(RMSE)$ değerleri aynı olmuştur ve başarıdan ödün verilmemiştir.

5 Sonuç ve Gelecek Çalışmalar

Test maliyeti fonksiyonunun pratik bir şekilde ifade edilebilmesi, gerek kalite kontrol sürecinin planlanabilmesi gerekse maliyeti dikkate alan kapsayan dizilerin hesaplanabilmesi için önemlidir. Test maliyeti fonksiyonlarının yazılım geliştirenler tarafından sağlıklı ve hatasız tanımlanamayacağı öngörüldüğünden maliyet fonksiyonlarını otomatik keşfedecek yöntemler geliştirilmiştir. Genelleştirilmiş lineer regresyon ve eleme tasarımları ile hesaplanan maliyet modelleri R-kare ölçüm metriği ile değerlendirilmiş ve maliyet hesaplamasında oldukça başarılı sonuçlar (sırasıyla ortalama 0.92 ve 0.99) elde edilmiştir.

Deneylerin sonuçları eleme tasarımlarının maliyeti modellemede kapsayan dizilerle hesaplanan genelleştirilmiş lineer regresyon modellerinden daha başarılı olduğunu göstermektedir. Bunun yanında eleme tasarımlarının boyutlarının lineer regresyon modellerin hesaplanmasında çalıştırılan kapsayan dizilerden yaklaşık 5 kat daha büyük olduğu görülmüştür. Bu demektir ki eğer yazılım mühendisleri bu yaklaşık 5 kat maliyeti istemezlerse, daha az başarılı olan ama ortalama R-karesi 0.9'dan büyük olan lineer regresyon modellerini de tercih edebilirler.

Yazılım mühendislerinin seçtiği konfigürasyon parametreleri her zaman doğru olmayabilir. Bu sebepten gelecekte eleme tasarımlarının lineer regresyon modelleri hesapladığımız çalışmamızda [9] kullandığımız daha büyük reel konfigürasyon uzayları üzerinde oluşturulması planlanmaktadır. Ayrıca bu parametrelerin otomatik olarak bir alt kümesinin seçilmesi içn kapsayan diziler ile oluşturulan lineer regresyon modellerinin kullandıkları parametrelerin eleme tasarımları için bir nevi parametre seçimi yöntemi olarak kullanılması da planlanmaktadır.

Teşekkürler

Bu araştırma TÜBİTAK tarafından desteklenmektedir (Proje No: 113E546).

Kaynaklar

1. D. M. Cohen, S. R. Dalal, M. L. Fredman, ve G. C. Patton. The AETG system: an approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, 23(7):437–44, 1997.
2. T. Berling ve P. Runeson. Efficient evaluation of multifactor dependent system performance using fractional factorial design. *IEEE Transactions on Software Engineering*, 29(9):769–781, 2003.
3. G. E. P. Box, W. G. Hunter, ve S. J. Hunter. *Statistics for experimenters: An introduction to design, data analysis, and model building.* New York: Wiley, 1978.
4. C. Yilmaz, S. Fouche, M. Cohen, A. Porter, G. Demiroz, ve U. Koc. Moving forward with combinatorial interaction testing. *Computer*, 47(2):37–45, Feb 2014.
5. M. H. C. F. Jeff Wu, *Experiments: Planning, Analysis, and Parameter Design Optimization*, Wiley, 2000.
6. I. Dunietz, W. K. Ehrlich, B. Szablak, C. L. Mallows, ve A. Iannino. Applying design of experiments to software testing: experience report. In *Proc. of the 19th Intl. Conference Software Engineering*, 205–215, ACM, 1997.
7. G. Demiroz ve C. Yilmaz. Cost-aware combinatorial interaction testing. In *Proc. of Fourth Int. Conf. on Advances in System Testing and Validation Lifecycle*, 2012.
8. G. Demiroz. Cost-aware combinatorial interaction testing (doctoral symposium). In *Proc. of the Int. Symp. on Software Testing and Analysis*, 440–443. ACM, 2015.
9. G. Demiroz ve C. Yilmaz. Towards Automatic Cost Model Discovery for Combinatorial Interaction Testing. In *Proc. of the 5th Int. Workshop on Combinatorial Testing (IWCT 2016)*, Chicago USA, April 2016.
10. G. Demiroz ve C. Yilmaz. Using simulated annealing for computing cost-aware covering arrays. *Applied Soft Computing*, available online August 2016.
11. D. R. Kuhn ve M. J. Reilly. An investigation of the applicability of design of experiments to software testing. In *Software Eng. Workshop*, 91–95. IEEE, 2002.
12. D. S. Hoskins, C. J. Colbourn, and D. C. Montgomery. D-optimal designs with interaction coverage. *Journal of Statistical Theory and Practice*, 3(4):817–830, 2009.
13. JMP Statistical Discovery Software from SAS, 2014. `http://www.jmp.com/`.
14. D. C. Montgomery, G. C. Runger, N. F. Hubele, *Engineering Statistics*, John Wiley & Sons, 2009.
15. C. Nie ve H. Leung. A survey of combinatorial testing. *ACM Computing Surveys*, 43:11:1–11:29, February 2011.
16. D. Kleinbaum, L. Kupper, A. Nizam, ve E. Rosenberg. *Applied regression analysis and other multivariable methods.* Cengage Learning, 2013.
17. S. R. Kenett ve Z. Shelemyahu, *Modern Industrial Statistics: The Design and Control of Quality and Reliability*, Cengage Learning, 1998.
18. C. Yilmaz and M. B. Cohen and A. Porter. Covering Arrays for Efficient Fault Characterization in Complex Configuration Spaces. *IEEE Transactions on Software Engineering*, 31(1):20–34, 2006.
19. C. Yilmaz, A. Porter, A. S. Krishna, A. M. Memon, D. C. Schmidt, A. S. Gokhale, ve B. Natarajan. Reliable effects screening: A distributed continuous quality assurance process for monitoring performance degradation in evolving software systems. *IEEE Transactions on Software Engineering*, 33(2):124–141, 2007.
20. Yilmaz, C. Test Case-Aware Combinatorial Interaction Testing. *IEEE Transactions on Software Engineering*, 39(5):684–706, 2013.