Yazılım test edilebilirliği: bir sistematik literatür haritalaması

Ebru İrge Hanoğlu[1], Ayça Tarhan[1], Vahid Garousi[1, 2]

1: Yazılım Mühendisliği Araştırma Grubu (HUSE)
Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi, Ankara
2: Maral Yazılım Mühendisliği Danışmanlık ve Ar-Ge Corp., Calgary, Kanada

{ebru.hanoglu, atarhan, vahid.garousi}@hacettepe.edu.tr

Özet: Yüksek kaliteli yazılımlar hem geliştiriciler hem kullanıcılar tarafından her zaman istenen bir sonuçtur. Çok boyutlu bir kavram olan yazılım kalitesi içsel ve dışsal pek çok faktör tarafından etkilenir. Test edilebilirlik yazılımın kalitesini etkileyen en önemli faktörlerden biridir. Bir yazılımın test edilebilirlik düzeyi ne kadar yüksekse, test eforu ve maliyeti o kadar düşük olacak; sonuçta ise güvenilir ve kaliteli ürünler ortaya çıkacaktır. Ne yazık ki test edilebilirlik yazılım ürünlerinin içsel bir özelliği olmadığı için doğrudan ölçülmesi mümkün değildir. Bu nedenle literatürde test edilebilirliği ölçmek adına pek çok öneriler ortaya konmuştur. Ölçümlerin çoğu kaynak kodları üzerinden gerçekleştirilmekte, ancak kodlama tamamlandıktan sonra gerek analizden kaynaklanan hataların gerek kodlama hatalarının giderilmesi daha maliyetli ve karmaşık olmaktadır. Bu makalenin amacı test edilebilirliği tahmin eden ya da ölçmeyi sağlayan modellerin varlığı üzerine bir literatür haritalaması gerçekleştirmektir.

Anahtar sözcükler: yazılım testi, yazılım test edilebilirliği, tahminleme modeli, sistematik literatür haritalaması

Software testability: a systematic literature mapping

Abstract. High-quality software is a result desired by all users. Software quality is a multidimensional concept and is influenced by many external and internal factors. Testability is one of the most important factors affecting the quality of the software. The higher the level testability of a software, lower the testing effort and cost will be; therefore more reliable and higher quality products will ultimately emerge. Unfortunately we cannot directly measure the testability of software since it is not an intrinsic property of the software. Therefore, many proposals are set forth in the literature in order to measure the testability. Most measurement proposals are carried out based on the source code, but we note that it is more costly and complex to eliminate errors resulting from the analysis or implementation phases. The purpose of this article is to perform a literature mapping on the existing models and techniques that are targeted to measuring or predicting the testability

Keywords: .Keywords: Software testing, software testability, systematic mapping, systematic literature review

1 Giriş

McKinsey & Company ve Oxford Üniversitesi işbirliği ile 2010 yılında 5,400 bilişim projesi üzerinde gerçekleştirilen bir çalışmanın sonuçlarına göre, yazılım projelerinin %45'i planlanan bütçeyi; %7 si belirlenen geliştirme süresini aşmakta; %56'sı ise beklenen gereksinimlerin çok azını karşılayarak tamamlanabilmektedir [1]. Yazılım projelerinde görülen bu başarısızlıklar yalnızca zaman kaybına değil, çok büyük mali kayıplara da neden olmaktadır. Bu büyük kayıpların önüne geçmek için alınacak ilk tedbirlerden biri yazılımda kalitenin artırılmasına yönelik çalışmalardır. Yazılım kalitesi geliştiriciye ya da kullanıcıya göre değişebilen ve birçok boyutu olan bir kavramdır. Juran'a göre [2] kalite kullanıma uygunluk olarak tanımlanırken, Crosby [2] kaliteyi sistemin gereksinimleri karşılama düzeyi ile ölçer. Bazı durumlarda yazılımın eksiksiz ve hatasız olması kritikken, bazı durumlarda kullanım kolaylığı kalitenin ölçüsü olabilmekte ve yazılımda bulunan hataların bakım evresinde giderilmesi sorun teşkil etmemektedir.

ISO/IEC 9126-1 standardına [3] göre yazılım kalitesi; içsel kalite özellikleri, dışsal kalite özellikleri, kullanımdaki kalite olarak modellenmektedir. Bu modele göre yazılımın kalitesi işlevsellik, güvenilirlik, kullanılabilirlik, verimlilik, bakım kolaylığı gibi özellikleri üzerinden değerlendirilmektedir. Kalite açısından önemli bir kriter olan bakım kolaylığı "yazılımın değişiklik veya düzeltme isteklerine adaptasyon yeteneği" olarak tanımlanmaktadır [4] ve bu makalede üzerinde durulan test edilebilirlik özelliği, bakım kolaylığının bir alt kategorisi olarak tanımlanmıştır.

Yazılım testi, yazılımdaki hataları bulmak, riskleri tespit etmek ve mevcut uygulamayı tanımlanan en yüksek kalite seviyesine ulaştırmak için yapılan testler bütünüdür. Yazılım test faaliyetleri ile yazılımda yer alan eksiklikler ve hatalar yazılım geliştirme sürecinin erken fazlarında fark edilir ve bunların giderilmesi ile kaliteli, kullanıcıyı ve geliştiriciyi daha çok tatmin eden ürünler ortaya çıkar. Daha az maliyetle daha kaliteli ürünler ortaya koymak için yazılımın test edilebilirliğini artırmak gerekmektedir. [5]

Yazılım test edilebilirliği, bir yazılım ürününün test faaliyetlerini desteklemesinin ölçüsüdür. Bir yazılımın test edilebilirlik düzeyi ne kadar yüksekse, test eforu ve maliyeti o kadar düşük olacaktır [4].

Test edilebilirlik yazılım ürünlerinin içsel bir özelliği olmadığı için doğrudan ölçülmesi mümkün değildir. Bu nedenle yazılımın test edilebilirliğini ölçmek için metrikler, modeller ve metotlar önerilmiştir. Bu makalenin amacı test faaliyetlerini kolaylaştırmak adına, literatürde mevcut olan, yazılım test edilebilirliğini tahmin etmeye yönelik modelleri gözden geçirmektir.

Makalenin ilerleyen kısımlarında; Bölüm 2'de ilgili çalışmalar verilmiştir. Bölüm 3'de bu çalışmada kullanılan literatür haritalama yöntemi tanımlanmıştır. Bölüm 4 literatürdeki çalışmaların bulgularının belirlediğimiz araştırma soruları kapsamında analizini içermektedir. Bölüm 5'de bu çalışmamızın sonuçları ve gelecekte ileriye yönelik yapılabilecek çalışmalar anlatılmıştır.

2 Bağlam ve ilgili çalışmalar

Kanıta-dayalı yaklaşım ilk olarak tıp alanında kullanılmış bir araştırma yaklaşımıdır ve daha sonra pek çok bilimsel alanda kullanılmaya başlanmıştır. Bu yaklaşımı ilk kez 2004 yılında Kitchenham ve arkadaşları "Kanıta-dayalı yazılım mühendisliği" [6] olarak yazılım mühendisliği alanına uyarlamışlardır. Bu bağlamda kanıtlar, özelleştirilmiş bir konu ya da soru üzerine gerçekleştirilmiş en kaliteli çalışmaların sentezi ile tanımlanır. Bu sentezi gerçekleştirmenin temel metodu sistematik literatür taramaları gerçekleştirmektir.
Sistematik literatür taramaları, araştırma konusuna yönelik katkı sağlayan birincil çalışmaları değerlendiren ikincil bir çalışmadır. Daha ayrıntılı bir tanım yapmak gerekirse; sistematik literatür taramaları belirli bir soruya yanıt ya da probleme çözüm oluşturmak için, o alanda yayınlanmış tüm çalışmaların kapsamlı bir biçimde taranarak, çeşitli dâhil etme ve dışlama kriterleri kullanarak ve araştırmaların kalitesi değerlendirilerek hangi çalışmaların derlemeye alınacağının belirlenmesi, derlemeye dâhil edilen araştırmalarda yer alan bulguların sentezlenmesidir. Tablo 1, bu çalışma ile ilgili yapılmış ikincil çalışmaları göstermektedir.

Tablo 1. İlgili çalışmalar (bu alanda yapılan ikincil çalışmalar)

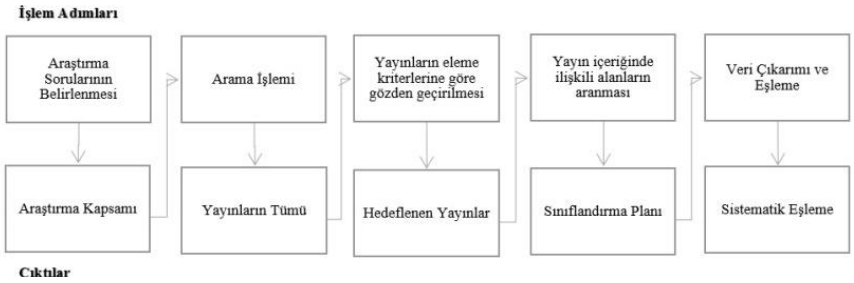
Makale Adı	**Yılı**	**Ref.**	**Çalışma-nın tipi**	**Birincil Çal. Say.**
Design for testability: a survey	1982	[7]	Normal survey	109
A survey of reliability, maintainability, supportability, and testability software tools	1991	[8]	Normal survey	233
Survey of source code metrics for evaluating testability of object oriented systems	2010	[10]	Normal survey	95
Measuring testability of object oriented design: a systematic review	2014	[11]	SLR	29
Testability and software robustness: a systematic literature review	2015	[12]	SLR	38
Testability and software performance: a systematic mapping study	2016	[13]	SM	34

3 Literatür haritalama metodu

Literatür taraması kapsamında incelenecek makalelere karar verilmesi adına öncül bir sistematik haritalama çalışması gerçekleştirilmiştir. Bu çalışma aşağıda belirtilen adımlar izlenerek gerçekleştirilmiştir ve akış diyagramı Şekil 1 ile gösterilmiştir:

- Haritalama sorularının tanımlanması
- Konuyla ilgili yayınlara ulaşmak için, elektronik veri tabanlarında ve diğer kaynaklarda aramaların yapılması ve incelenecek yayınların tespit edilmesi
- İçerme (include) ve dışarıda bırakma (exclude) kriterlerinin belirlenmesi
- Belirlenen kriterlere göre yayınların seçilmesi
- Yayınların içerikleri taranarak, sınıflandırmak için kullanılacak olan yayın tarihlerinin, ilişkili alanların, çalışma türlerinin ve yayın türlerinin saptanması

Sistematik literatür taramaları, 3 ana fazda özetlenebilecek ayrık faaliyetleri kapsar. Bu fazlar planlama, gerçekleştirme ve raporlama aşaması olarak adlandırılır. Sistematik eşleme çalışması ile incelenecek makaleler belirlendikten sonra Kitchenham'ın "Procedures for Performing Systematic Reviews" çalışmasında belirtilen prosedür takip edilerek literatür taraması gerçekleştirilmiştir.



Şekil 1. Sistematik haritalama süreci ve çalışma adımları

3.1 Haritalama soruları

Bu literatür tarama çalışması aşağıdaki Haritalama Sorularını (HS) yanıtlamak için tasarlanmıştır:

HS 1. Çalışmaların araştırmaya yönelik katkıları nelerdir? Çalışmaların kaç tanesi metot, teknik, model, araç veya süreç sunmaktadır?
HS 2. Çalışmalarda kullanılan araştırma yöntemleri nelerdir?
HS 3. Bir yazılımın test edilebilirlik düzeyi modeller kullanılarak nasıl tahmin edilebilir? Bu modellerin güvenilirlik düzeyleri nasıldır?
HS 4. En çok atıf sayısına sahip makaleler hangileridir?
HS 5. Yapılan çalışmaların endüstri/akademik olmalarına göre dağılımları nasıldır?

3.2 Kaynakların (birincil çalışmaların) arama ve seçimi

Haritalama sorularının belirlenmesinin ardından ilgili makalelere erişmek için anahtar sözcükler belirlendi ve bu anahtar sözcükler kullanılarak üç farklı dijital kütüphanede arama yapıldı. Konu ile alakalı olduğu düşünülen makaleler başlıklarına, özetlerine ve anahtar sözcüklerine bakılarak havuza dâhil edildi. Aramaların gerçekleştirildiği anahtar sözcükler şu şekilde gruplanmıştır:

```
model AND {predict OR estimate OR estimation} AND {software testability}
```

Tablo 2. Kaynakların arama aşaması sonucunda literatür taraması için seçilen kaynaklar

Kaynak	Makale Sayısı	Periyot
Google Scholar	642	1996 - 2015
ACM	4	1996 - 2015
IEEE Xplore	43	1996 - 2015

3.3 Dâhil etme ve dışlama kriterleri

İkinci aşamada makaleler dâhil etme ve dışlama kriterleri göz önünde bulundurularak değerlendirildi ve sayısı 29'a düşürüldü. Bu çalışma boyunca göz önünde bulundurulan dâhil etme ve dışlama kriterleri her bir makale için şu şekilde belirlendi:

- Tez şeklinde ya da dergide, konferansta ya da workshoplar kapsamında yayınlanmış olmalı,
- İngilizce olarak yazılmış olmalı,
- Konu ile doğrudan alakalı olmalı,
- Sunularına dair geçerli bir kanıt sunmalı,
- Son 20 yıl içerisinde yayınlanmış olmalı
- Bir kestirim modeli önermeli

Yapılan çalışmaların büyük çoğunlunun test edilebilirliği kaynak kod üzerinden ölçmek üzerine olduğu görüldü ve geliştirme aşamasına gelmeden test edilebilirliği ölçen çalışmaların kodun test edilebilirliğini tahmin ettiği düşünülerek havuza dâhil edildi.

3.4 Çalışmaların sınıflandırılması

Ayrıntılı incelenecek makaleler belirlendikten sonra aşağıda belirtilen 4 grupta toplandı:

- A: Doğrudan test edilebilirliği tahmin etmek için modeller
- B: Test edilebilirliği yazılım geliştirme sürecinin erken fazlarında ölçmeyi öneren modeller (diagram veya doküman üzerinden)
- C: Test edilebilirliği inşa etmeyi öneren modeller
- D: Model önermeyen ancak konuya katkı sağlayan makaleler

A kategorisinde yer alan makale sayısının çok az olması nedeniyle çalışmaya B kategorisinde yer alan makaleler de dahil edildi ve bu modeller haritalama sorusu 3'ün sonuçlarında özetlendi. Hangi kategoride kaç makalenin yer aldığı Tablo 3'te özetlendi.

Tablo 3. Kategorilere göre makale sayılarının dağılımı

Kategori	Makale Sayısı
A	6
B	2
C	2
D	19

3.5 Kalite değerlendirmesi

Havuza eklenen birincil çalışmaların kalite değerlendirmeleri Tablo 4'de verilen kontrol listesi kullanılarak gerçekleştirildi. Her soru için çalışmanın kalitesi 'evet', 'kısmen' veya 'hayır' olarak değerlendirildi. Cevapların ağırlığı sırasıyla 1, 0.5, ve 0 olarak belirlendi.

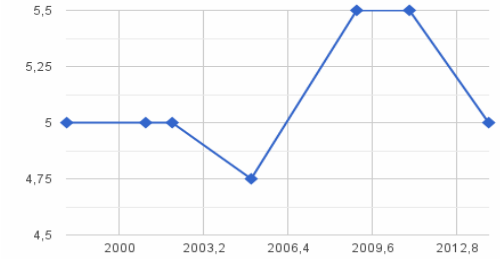
A ve B kategorisine ait çalışmalar yukarıda belirtilen 7 soru üzerinden

değerlendirilerek kalite değerlendirmeleri gerçekleştirildi.

Tablo 4. Kaynakların çalışmaların kalite değerlendirmesinin yapılması

Sorular	**Skor**
Çalışmanın amacı/araştırma soruları net bir şekilde belirlenmiş mi?	e/k/h
Çalışmada ilişkili çalışmaların sonuçları değerlendirilmiş mi?	e/k/h
Çalışmada gelecek çalışmalar için önerilerde bulunulmuş mu?	e/k/h
Önerilen modelin varsayımları açıkça izah edilmiş mi?	e/k/h
Önerilen modelin yapısı tam olarak tarif edilmiş mi?	e/k/h
Model endüstri tabanlı bir vakada denenmiş mi?	e/k/h
Çalışmada modelin uygulanması ile elde edilen bulgular belirtilmiş mi?	e/k/h

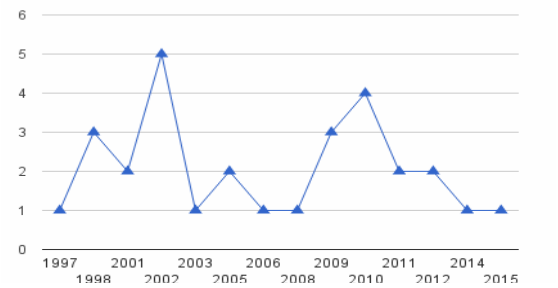
Çalışmalara ait kalite değerlendirme sonuçlarının yıllara göre dağılımı Şekil 2'de gösterildi. İncelenen çalışmaların kalite değerlendirmelerinin ortalaması 5,1/7 olarak tespit edildi.



Şekil 2. Yıllara göre çalışmaların kalite değerlendirme sonuçları

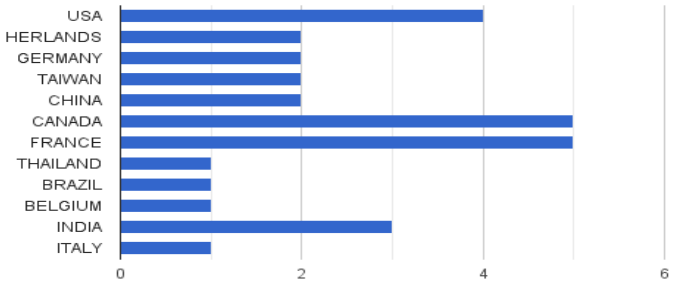
3.6 Birincil çalışmalar

İncelenecek birincil çalışmalara karar vermek adına sistematik haritalama çalışmasına başlandı. Belirlenen anahtar sözcüklerle arama yapıldığında toplam 689 makale bulundu. Konuyla alakalı olarak gözden kaçırılmış makalelerin olmaması içini havuzda yer alan ve en çok atıf sayısına sahip makalelerin referansları veya bu makaleleri referans olarak kullanan çalışmalar incelenerek havuz genişletildi. Havuzda yer alan makaleler başlık, özet ve anahtar sözcükleri ile incelenerek ilk gözden geçirme gerçekleştirildi ve makale sayısı 82'ye düşürüldü. Daha sonra yapılan incelemeye makalelerin giriş ve sonuç kısımları da dâhil edilerek birincil çalışmaların sayısı 29 olarak belirlendi.
Ardından bu 29 makale yayınlandıkları yıllara göre sınıflandırıldı. Bu makalelerin yıllara göre dağılımı Şekil 3'de gösterilmiştir. Test edilebilirliği tahmin etmeye yönelik çalışmalara belirli yıllarda yoğunlaşılmış olunsa da, son yıllarda popülaritesini kaybetmiş olduğu görülmektedir.



Şekil 3. Yıllara göre çalışmaların dağılımı

Şekil 4 ilk yazarlarının ülkesine göre çalışmaların dağılımını göstermektedir. Seçilen son çalışmaların 12 farklı ülkede gerçekleştirildiği görülmektedir. Çalışmaların büyük çoğunluğu Kanada ve Fransa'da gerçekleştirilmiştir.



Şekil 4. Çalışmaların yapıldığı ülke dağılımı

3.7 Veri çıkarımı ve sistematik harita

Her çalışma için temel özellikler incelenerek özellikler belirlenmiş ve gruplanarak Tablo 5'te gösterilmiştir. 'Soru' başlıklı sütun, özelliklerin ve tanımlamaların ilişkili olduğu araştırma sorusunun numarasını göstermektedir.

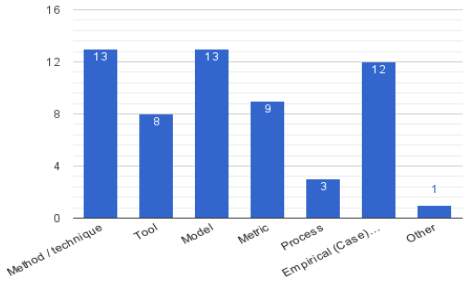
Tablo 5. Veri çıkarımı aşamasında belirlenen özellikler (sistematik harita)

Soru	Özellik	Tanımlar
HS1	Katkı türü	Metot, araç, teknik, metrik, deneysel çalışma, diğer
HS2	Araştırma metodu	Çözüm önerisi (solution proposal), geçerleme çalışması (validation research), değerlendirme çalışması (evaluation research), deneyim makalesi (experience paper), diğer
HS3	Tahminleme modeli	
HS4	Atıf sayıları	Atıf Sayısı: Sayı
HS5	Çalışmanın özelliği	Yazarların Çevresi: {Akademik, Endüstri, Kombinasyonel}

4 Sonuçlar

4.1 HS 1-Çalışmaların araştırmaya katkı tipleri

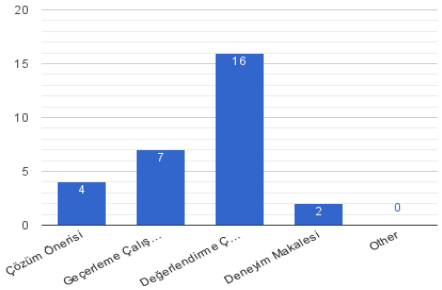
Final havuzunda bulunan 29 makale katkı tiplerine göre sınıflandırıldı ve elde edilen sonuçlar aşağıdaki şekillerde gösterildi. Bu kısımda bir çalışma birden fazla özelliğe sahip olabildiği için toplam tip sayısı makale sayısından fazla olmaktadır. Şekil 5'te görüldüğü gibi çalışmalarda en çok metot/teknik ve model önerilmiştir. Çalışmaların 12 tanesi deneysel çalışmalarla desteklenmiş ve sunularının güvenilirliği artırılmıştır.



Şekil 5. Çalışmaların katkı türleri

4.2 HS 2-Araştırma yöntemi

Final havuzunda bulunan 29 makale, araştırma yöntemlerine göre sınıflandırıldı. Bu kısımda bir çalışma yalnızca bir araştırma yöntemine sahip olabilmektedir. Şekil 6'da görüldüğü gibi çalışmaların büyük çoğunluğu değerlendirme çalışması olarak gerçekleştirilmiştir. 7 adet geçerleme çalışması mevcut iken, 4 çalışma konuya çözüm önerileri ile yaklaşmıştır.



Şekil 6. Araştırma yöntemleri

4.3 HS 3-Tahminleme modeli

Yapılan taramalar sonucunda test edilebilirliği tahmin etmeye yönelik 6 modelin

öne çıktığı görüldü. Test edilebilirliği anlamak için akış diyagramlarını ve UML modellerini de kullanan 2 çalışma incelenerek bu kısımda sunuldu.
Çalışmalarda sunulan modeller, uygulandıkları test alanları (test domains), modelin kullanım amacı ve kalite değerlendirmeleri Tablo 6 ile gösterildi. Tabloda bahsedilen ölçme metotları "A Component Testability Model for Verification and Measurement" [14] makalesinde verilen metotlar referans alınarak düzenlenmiştir.
"Testability Growth Model" [15] Markov'un zincir-tabanlı modeli temel alınarak geliştirilmiş bir test-düzelt-test modelidir. Test-düzelt-test modelinin mantığı, bir hata bulunduğunda düzeltilene kadar geliştirmeye ara verilmesi ve hata düzeltildikten sonar devam edilmesi mantığına dayanmaktadır. Değerlendirmeler 'design for testability ' (DFT) metriklerinin hatalar arasında beklenen ve aslında elde edilen sonuçları Bayes yaklaşımı ile değerlendirilerek yapılmaktadır. Önerilen model 3 farklı simülasyon üzerinde uygulanmış ve elde edilen sonuçlar modelin uygulanabilir olduğunu göstermiştir.
"Metric-Based Testability Model for Object-Oriented Design (MTMOOD)" [16] sınıfların test edilebilirliğini dizayn aşamasında sınıf diyagramlarının bazı nesne yönelimli tasarım özelliklerini analiz ederek ölçmeyi hedeflemektedir. Bunların başlıcaları kalıtım, sarmalama, bağımlılık gibi kriterlerdir. Model pek çok proje üzerinde defalarca denenmiş ve elde edilen sonuç eksiklikleri ve avantajları ile ayrıntılı olarak listelenmiştir. Gelecek çalışmalarda model başka projeler üzerinde defalarca denenerek genelleştirilebilir sonuçlara ulaşmak hedeflenmektedir.
"The Model of Testability Measurement" [17] yanlış tanımlanan değişkenler/sabitler, sabit olarak tanımlanan değişkenler, değişken olarak tanımlanan sabitler ve yanlış kullanılan operatörler üzerinden test edilebilirliği hesaplamayı amaçlamaktadır. Model basit bir örnek program üzerinde denenmiş ve Voas'ın PIE modeli ile elde edilen sonuçlar birbirine yakın bulunmuştur. Gelecek çalışmalarda modelin iyileştirilmesi hedeflenmiştir.
"Prediction Model For Evolutionary Testability" [18] model içerisinde tanımlanan evrimsel test edilebilirlik metriklerinin ve gelecek varlıkların (future entitiy) matematiksel modellerle çözümlenerek analiz edilmesine dayanır. Model gerçek zamanlı sistemler için düşünülmüştür. Model laboratuar ortamında denenmiş olup çıkan sonuçlar güvenilirliğini yüksek gösterse de endüstriyel ürünlerde denenip sonuçların değerlendirilmesi gelecek çalışmalara bırakılmıştır
"A Qualitative Model of Run-Time Testability" [19] test edilebilirliği, çalışma anında test edilebilirliği etkileyen ana faktörler göz önünde bulundurularak, tanımlanan metrikler kullanılarak grafikler ya da akış diyagramları üzerinden hesaplamayı amaçlamaktadır. Model iki farklı bileşen tabanlı sistem üzerinde denenmiş ve sonuçlar modelin uygulanabilir olduğunu göstermiştir.
"A Component Testability Model" [20] geliştirme süresince bileşenlerin geçerleme ve ölçme faaliyetlerini desteklemesini sağlamak üzerine tasarlanmıştır. Benzer çalışmaların aksine modelin tasarlanmasındaki amaç gereksinimden teste kadar geçerleme çalışmalarını destekleyen bileşenler inşa edilmesini sağlamaktır. Model ile gerçekleştirilen durum çalışmaları, modelin geçerliliğinin yüksek olduğunu ortaya koymaktadır.

"UML-Based Models" [21] test durumlarının otomatik yaratılmasına, uygulanmasına ve değerlendirilmesine dayanan bir modeldir. Bu şekilde test durumlarının oluşturulma kolaylığının ve güvenirliğinin artırılması hedeflenmiştir. Manuel test yöntemleri ile UML tabanlı modeller kullanılarak oluşturulan otomatik test durumları kıyaslanmış ve sonuçta modelin faydalı olduğu görülmüştür.

Tablo 6. Seçilen makalelerden özetler

		Ölçme Yöntemi				**Kullan. Amacı**		
Çalışmanın Adı	**Önerilen**	**Prog.-based**	**Model-based**	**Dependability asses.**	**Metrics**	**Ölçme**	**Tahmin**	**Kalite değ.**
A testability growth model and its application [15]	Testability Growth Model			√			√	5
An empirical analysis of a testability model for object-oriented programs [16]	(MTMOOD)	√				√	√	5
An analytic software testability model [17]	The Model of Testability Measurement	√			√	√		5,5
A prediction system for evolutionary testability applied to dynamic execution time analysis [18]	Prediction Model For Evolutionary Testability			√	√	√	√	5
A Model for the Measurement of the Runtime Testability of Component-based Sys. [19]	Qualitative Model of Runtime Testability	√					√	6,5
A Component Testability Model for Verification and Measurement [20]	A Component Testability Model			√			√	6
A UML-based approach to system testing – Briand [22]	TOTEM System Test Meth-			√			√	3,5
A UML-based approach to system testing – Hartman [21]	UML-Based Models			√			√	4,5

"TOTEM System Test Methodology" [22] use-case diyagramlarını ve tanımlamalarını, her use-case için akış ve bileşen diyagramlarını, sınıf diyagramlarını ve her sınıf ve metot için oluşturulan veri sözlüğünü içeren bir modeldir. Test aktivitelerini bir sistematiğe oturtmayı ve onları otomatize ederek kapsamının genişlemesini sağlamayı hedeflemektedir. Ne yazık ki modeli sınamak ilerleyen çalışmalara bırakılmıştır. Önerilen modellerin güvenlikleri kalite değerlendirmesinin sonuçları ile aynıdır.

4.4 HS 4-En çok atıf alan makaleler

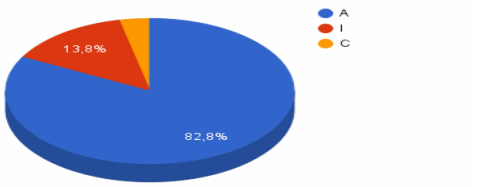
Havuzda bulunan makaleler atıf sayılarına göre sınıflandırılmış ve en çok dikkat çeken 5 makale Tablo 7'da gösterilmiştir.

Tablo 7. En çok atıf sayısına sahip 5 makale

Çalışma adı	Atıf sa.
A UML- based Approach to System Testing - Briand	282
A UML-based Approach to System Testing - Hartmann	69
A Prediction System for Evolutionary Testability Applied to Dynamic Execution Time	28
A Model for the Measurement of the Runtime Testability of Component-based Systems	30
A Component Testability Model for Verification and Measurement	43

4.5 HS 5-Endüstri veya akademiden gelen makaleler

Çalışmaların yazarlarına göre akademik-endüstriyel ya da tümleşik olduklarına karar verilmiş ve sonuçlar Şekil 7'de gösterilmiştir. Şekil'de görüldüğü üzere bu konu hakkındaki çalışmalar genel olarak akademik kulvarda yürütülmektedir.



Şekil 7. Çalışmaların kaynağı (A: Akademik, I: Endüstriyel, C: Tümleşik)

5 Sonuçlar ve gelecek çalışmalar

Anahtar sözcüklerle arama yapıldığında farklı kaynaklardan toplam 689 makale bulundu. Gözden kaçırılmış önemli makalelerin olmaması için en çok atıf sayısına sahip makalelerin referansları ve onu referans alan makaleler incelenerek havuz genişletildi. Daha sonra başlık, özet ve anahtar sözcükleri incelenerek ilk gözden tarama gerçekleştirildi ve final havuzu sonuç kısımları ve referansları dikkatle incelenerek makale sayısı 82'ye indirildi. Daha sonra incelemeye giriş ve sonuç kısımları da dâhil ederek 9 makale ilişkili çalışma olarak değerlendirildi ve birincil çalışmaların sayısı 29 olarak belirlendi ve bu çalışmalar ayrıntılı olarak incelendi.Gelecek çalışmalar adına; bu çalışmayı geliştirerek daha derin bir sistematik literatür tarama (SLR) yapmayı planlıyoruz.

Kaynaklar

1. Michael Bloch, Sven Blumberg, and Jürgen Laartz, "Delivering large-scale IT projects on time, on budget, and on value", http://www.mckinsey.com/business-functions/business-technology/, Last accessed: June 15, 2016
2. Kat Kadian-Baumeyer, "Deming, Juran & Crosby: Contributors to Total quality management (TQM)", http://study.com/academy/lesson/deming-juran-crosby-contributors-to-tqm.html, Last accessed: June 15, 2016
3. ISO/IEC 9126-1: Information Technology - Software Product Quality - Part 1: Quality Model. ISO/IEC JTC1/SC7/WG6 (1999)

4. Ural Erdemir, Umut Tekimn ve Feza Buzluca, “Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi Object Oriented Software Metrics and Software Quality”
5. Nupul Kukreja, “Measuring Software Maintainability” Last accessed: June 16, 2016, https://quandarypeak.com/2015/02/measuring-software-maintainability/, Last accessed: June 15, 2016
6. Barbara A. Kitchenham, Tore Dyba, and Magne Jorgensen. "Evidence-Based Software Engineering", Proceedings of the International Conference on Software Engineering (ICSE), pp. 273-281, 2004
7. T. W. Williams and K. P. Parker. 1982. Design for Testability A Survey. IEEE Trans. Comput. vol. 31, no. 1, pp. 2-15, 1982
8. Caroli, Joseph A, "A Survey of Reliability, Maintainability, Supportability, and Testability Software Tools", US Military Tech. Report, Accession Number : ADA236148, 1991
9. Fu Jianping, Liu Bin and Lu Minyan, "Present and future of software testability analysis", International Conference on Computer Application and System Modeling , pp. V15-279-V15-284, 2010
10. Muhammad Rabee Shaheen, Lydie Du Bousquet, "Survey of source code metrics for evaluating testability of object oriented systems", Laboratoire d'Informatique de Grenoble, Tech. Report, RR-LIG-005, 2010
11. Mahfuzul Huda, D.S. Arya, Dr. M. H. Khan, "Measuring Testability of Object Oriented Design: A Systematic Review", International Journal of Scientific Engineering and Technology, Volume 3, Issue.10, pp. 1313-1319, 2014
12. M. M. Hassan, W. Afzal, M. Blom, B. Lindström, S. F. Andler and S. Eldh, "Testability and Software Robustness: A Systematic Literature Review," 2015 41st Euromicro Conference on Software Engineering and Advanced Applications, Funchal, pp. 341-348, 2015
13. Hassan, M. Mahdi and Afzal, Wasif and Lindström, Birgitta and Shah, Syed and Andler, Sten F. and Blom, Martin, Testability and Software Performance: A Systematic Mapping Study. ACM Symposium on Applied Computing, 2016
14. Jerry Gao and Ming-Chih Shih, “A component testability model for verification and measurement”, COMPSAC-W'05, pp. 211-218, 2005
15. Chenxu Zhao, Jing Qiu, Guanjun Liu, Kehong Lv and Krishina Pattipati, “A testability growth model and its application”, Systems, Man, and Cybernetics: Systems, IEEE Transactions on, On page(s): 524 - 534 Volume: 46, Issue: 4, April 2016
16. Aymen Kout, Fadel Toure, and Mourad Badri. "An empirical analysis of a testability model for object-oriented programs." ACM SIGSOFT Software Engineering Notes 36.4 (2011): 1-5.
17. Jin-Cherng Lin and Szu-Wen Lin. "An analytic software testability model."null. IEEE, 2002.
18. Hans-Gerhard Groß, "A prediction system for evolutionary testability applied to dynamic execution time analysis." Information and Software Technology 43.14 (2001): 855-862.
19. Adriana Gonzalez, Eric Piel and Hans-Gerhard Gross. "A model for the measurement of the runtime testability of component-based systems."Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on. IEEE, 2009.
20. Jerry Gao and Ming-Chih Shih. "A component testability model for verification and measurement." Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International. Vol. 2. IEEE, 2005.
21. Jean Hartmann, et al. "A UML-based approach to system testing." Innovations in Systems and Software Engineering 1.1 (2005): 12-24.
22. Lionel Briand and Yvan Labiche. "A UML-based approach to system testing."Software and Systems Modeling 1.1 (2002): 10-42.