Journal of Information Systems and Management Research
Bilişim Sistemleri ve Yönetim Araştırmaları Dergisi

http://dergipark.gov.tr/jismar

Araştırma Makalesi / Research Article

Yazılım Geliştirme Test Aktivitelerinde Kritik Başarı Faktörleri

Doğan YILDIZ[*,a]

[a,*] *Türk Havacılık ve Uzay Sanayii A.Ş. (TUSAŞ), Bilgi Teknolojileri Başkanlığı, ANKARA, TÜRKİYE*

MAKALE BİLGİSİ

Alınma: 06.06.2021
Kabul: 19.09.2021

Anahtar Kelimeler:
Test, Bilişim Sistemi, Yazılım Geliştirme, Kritik Başarı Faktörleri

Sorumlu Yazar
e-posta:
dyildiz2000@gmail.com

ÖZET

Test süreci, yazılımların işletime alınmadan önce herhangi bir hata var mı şeklinde farklı testlerin yapıldığı bir süreçtir. Hatasız bir bilişim sistemi için test aktivitelerine gereken önem verilmeli ve ilgili testlerin yapılmasına özen gösterilmelidir. Bu çalışmanın amacı, yazılım geliştirme projelerinin başarısını arttırmak için test süreçlerinde dikkat edilmesi gereken kritik hususlar hakkında bilgi sunmaktır. Bu çalışma kapsamında test süreci ile ilgili literatür taraması yapılmış ve ardından bu süreçteki yöntemler hakkında bilgi sunulmuştur. Uygulama kısmında farklı projelerdeki test süreci ve bu süreçteki test aktiviteleri, test senaryoları, testler sonucunda bulunan bulguların nasıl elimine edilebileceği ile ilgili kritik başarı faktörü ve dikkat edilmesi gereken hususlar ortaya konulmuştur.

Critical Success Factors in Software Development Test Activities

ARTICLE INFO

Received: 06.06.2021
Accepted: 19.09.2021

Keywords:
Testing, Information System, Software Development, Critical Success Factors

Corresponding Authors
e-mail:
dyildiz2000@gmail.com

ABSTRACT

The testing process is a process in which different tests are carried out to see if there are any errors before the software is put into operation. For an error-free information system, test activities should be given due importance and care should be taken to carry out the relevant tests. The purpose of this study is to provide information about critical issues that should be considered in testing processes in order to increase the success of software development projects. Within the scope of this study, a literature review about the test process was made and then information about the methods in this process was presented. In the application part, critical success factors and points to be considered about the test process in different projects and the test activities in this process, test scenarios, how to eliminate the findings found as a result of the tests are presented.

1. *GİRİŞ (INTRODUCTION)*

Bilişim sistemi geliştirme safhalarını analiz, tasarım, kodlama ve test şeklinde bölümlediğimizde, test safhası sistemin işletime verilmeden önce hatalara karşı son kontrollerinin yapıldığı ve özellikle kritik sistemler için çok önemli bir safhadır. Bu safhaya gereken önemin verilmemesi durumunda sistem, hatalı bir şekilde işletime verilebilir ve insan hayatını tehlikeye atacak sonuçlar doğurabilir. Buna rağmen bilişim sistemleri geliştirme projelerinde bu safhaya gereken zaman verilmemektedir. Bunun için bu safhada gerekli testlerin yapılması ve yazılımdaki hataların bulunarak düzeltilmesi ve tekrar test edilmesi en doğru yaklaşımdır. Özellikle yazılım geliştirme projesinde zaman bu safhadan önceki safhalar olan sistem analizi ve tasarımı ile kodlama safhalarında harcanmakta ve bu safhaya gelince de projedeki zaman baskısından dolayı yöneticiler tarafından bu safhanın çok hızlı geçilmesi istenebilmektedir. Bu baskı aynı şekilde çevik geliştirme projelerinde de yaşanabilmektedir. Çevik

yazılım geliştirme projelerinde de sprint (scrum yazılım geliştirme yönteminde 2 ila 4 hafta arasında yapılan koşu) içine alınan işin ilgili sprint içinde bitmesi gerektiğinden dolayı önceden planlanan yazılım geliştirme zamanı uzadıkça ve ilgili işin de sprint içinde bitirilme baskısından dolayı bazen çok etkili bir şekilde zaman ayrılmadan testler yapılmakta ve iş bu şekilde tamamlanarak işletime verilebilmektedir. Tüm bunların önüne geçmek ve test sürecine gereken önemi vermek için proje yöneticilerinin veya takımın (scrum veya şelale yazılım geliştirme takımının) test yapmanın kritikliğinin farkına varması, zaman planlarını test sürecinin dinamiklerini dikkate alarak yapması gerekmektedir. Yazılım geliştirme sürecinden hatalı bir şekilde çıkacak olan yazılımın işletime verildikten sonra çıkan hatalarının düzeltilmesinin daha da maliyetli olacağının bilinmesi ve ona göre tüm planlamaların dikkatli bir şekilde yapılması projenin başarısı için daha iyi olacaktır.

Bu çalışma ile hem araştırmacı hem de uygulayıcılar açısından yazılım geliştirme projelerindeki test sürecinde nelere dikkat etmeleri gerektiği konusunda bilgi sunulması hedeflenmiştir. Test sürecinde gözden kaçacak bir yazılım hatası yazılım işletime verildiğinde çok büyük sorunlar doğuracağı için bu süreçteki kritik başrı faktörlerinin dikkatli bir şekilde hem araştırmacı hem de uygulayıcılar açısından incelenmesi ve uygulanması önemlidir. Bilimsel alanda bu çalışma test sürecindeki kritik başarı faktörlerini işin teorisinde ve uygulamasında çalışanlara yol gösterici olacaktır. Yazılım geliştirme test sürecinde hatalar yakalanbilmekte midir? Yazılım geliştirme test süreci nasıl başarılı bir şekilde geçilebilir? Bu süreçte nelere dikkat etmeliyiz? gibi araştırmacıların sorularına cevap bulmaları amaçlanmıştır.

Bu çalışma kapsamında öncelikle literatür incelemesi yapılmış, ardından test yöntemleri ve süreci hakkında bilgiler verildikten sonra uygulama kısmında ise test sürecinde kritik başarı faktörlerine yer verilmiştir.

1.1. Literatür Taraması *(Literature Review)*

Bilişim sistemi yazılım geliştirme test aktiviteleri ile ilgili farklı birçok çalışma yer almaktadır. Basili ve Selby (1987) yapmış oldukları çalışmalarında kod okuma, fonksiyonel test ve yapısal test şeklinde üç farklı yazılım test stratejisini 42 ileri seviye öğrenci ile uygulamışlardır. Bunun sonucunda; 1) Profesyonel programcıların kod okuma ile daha fazla hata bulduklarını görmüşlerdir, 2) Bir grup ileri seviye öğrencilerde fonksiyonel test ile kod okuma aynı sonuç vermiştir, 3) Bir grupta ise bu üç yönteminde aynı hata bulma sonucunu verdiğini görmüşlerdir, 4) Yazılım testinin hata bulma sayısından farklı sayıda hata bulmuşlardır, 5) Kod okuma, önyüz hatalarının bulunmasında faydalı olmuştur, 6) Kontrol hataları fonksiyonel test soncunda elde edilmiştir, 7) Hataların yüzde kaçı hangi yöntem ile bulunduğunu sorduklarında, kod okuyan grubun daha doğru sonuç verdiğini görmüşlerdir [1]. Gelperin ve Hetze (1988) çalışmalarında dört adet test modeli üzerinden bilgiler sunmuşlardır. Bu dört model iki ana grupta toplanmıştır. İlk grup safha modelleridir. Bunun altında iki alt model vardır. Bunlar; 1) Gösteri: Yazılım sepesifikasyondaki özellikleri sağlaması, 2) İmha: Uygulama hatalarını tespit edilmesi şeklindedir. İkinci grup ise ömür devri modelleridir. Bunun altında ise yine iki alt model vardır. Bunlar; 1) Değerlendirme: Gereksinim, tasarım ve uygulama hatalarını tespit edilmesi, 2) Önleme: Gereksinim, tasarım ve uygulama hatalarını önlenmesi şeklindedir [2]. Whittaker (2000) çalışmasında test sürecini dört safhaya bölmüştür. Bunlar, yazılım ortamının modellenmesi, test senaryolarının seçilmesi, test senaryolarının koşulması ile değerlendirilmesi ve son olarak da test sürecinin ölçülmesidir [3]. Tassey (2002) yazılım kalitesi ile ilgili nitelikleri farklı bakış açıları ile neler olduğunu açıklamıştır. Bunun yanında yazılım kalite metrikleri ile ilgili de bilgiler aktarmıştır. Yazılım testini yazılımın önceden belirlenmiş olan kriterlere uygun olup olmadığının belirlenmesi için yapılan faaliyetler olarak tanımlamaktadır. Standart test işlemlerinin olmamasının etkilerini dört grupta toparlamıştır. Bunlar; kalitesizlik nedeni ile artan arızalar, yazılım geliştirme maliyetinde artış, verimsiz testler nedeni ile ürünün kullanıma sunuşunda gecikmeler ve piyasa işlem maliyetlerindeki artışlardır. Bunun yanında 1960 ve 1970'li yıllarda yazılım geliştirmedeki ağırlığın kodlama tarafında olduğunu ve bu ağırlığın giderek sistem analizi ve tasarımı ile test tarafına doğru kaydığını aktarmıştır [4]. Dustin (2003), etkili yazılım testi geliştirmenin 50 özel yolu adlı eserinde gereksinim safhasında, test planlama safhasında, test takımı, sistem yapısı, test tasarımı ve dokümantasyonu, birim test, otomatik test araçları, otomatik test, fonksiyonel olmayan test, testin yapılmasının yönetimi başlıkları altında önerilerde bulunmuştur [5]. Bertolino (2007) çalışmasında yazılım testi ile ilgili olarak araştırmaları incelediğinde geçmişteki başarılar, günümüzdeki zorluklar ve gelecekte yapılması gereken rüyalar olarak bilgiler sunmuştur. Başarılar kısmında, test kriterleri (test senaryoları), test kriterlerinin karşılaştırılması, nesne tabanlı test, bileşen tabanlı test, protokol testi ve güvenilirlik testidir. Gelecekteki işleri dört ana grup altında toparlamıştır. Bunlar; evrensel test teorisi, test tabanlı modelleme, yüzde yüz otomatik test ve verimliliği yüksek test mühendisliğidir. Zorluklar altında ise test yapan

personelin eğitimi, test desenleri, test yapma maliyeti, çevrimiçi test, özel test yaklaşımları, test için veri türetme, model tabanlı olmayan testler ve test etkililiğidir [6]. Orso ve Rothermel (2014) yapmış oldukları çalışmalarında 2000 yılından beri yazılım testi ile ilgili yapılan öncül çalışmalar ve gelecekte bu konuda fırsatlar ile ilgili yaptıkları anket çalışması ile ilgili bilgiler sunmuşlardır. Sonuçları dört kategoride ele almışlardır. Bunlar, otomatik test girdisi oluşturulması, test stratejileri, regresyon testi, deneysel çalışmalar ve desteklerdir [7]. Baran, Akar ve Aslay (2016), geleneksel yazılım geliştirme ve test süreci hakkında bilgiler sunmuşlardır. Ardından uç programlama (Extreme Programming [XP]), scrum, yalın yazılım geliştirme (Lean Software Development [LSD]), kanban, özellik güdümlü programlama (Feature Driven Development [FDD]), dinamik sistemler geliştirme yöntemi (Dynamic Systems Development Methodology [DSDM]) ve adaptif yazılım geliştirme (Adaptive Software Development [ASD]) çevik yöntemlerindeki test süreçleri hakkında bilgi sunmuşlardır [8]. Bu çalışmanın yanısıra Baran, Aslay ve Akar (2016) Geleneksel Yazılım Geliştirme Yöntemlerinde Test Problemleri adlı çalışmalarında ise geleneksel yazılım geliştirme yöntemlerinden şelale modeli, V model, Helezonik Model, W modelindeki test süreçleri ile ilgili bilgiler sunmuşlardır [9]. Arumugam (2019) çalışmasında ise yazılım kalitesini arttırmak için mevcut ve geliştirilmiş test tekniklerini tartışmıştır. Bu kapsamda beyaz, siyah ve gri kutu test teknikleri, bunun yanında yazılım yaşam döngüsünde ki test safhaları ve son olarak da test otomasyonu ve test odaklı geliştirme ile ilgili bilgiler sunmuştur [10]. Aydıner, Esen ve Özlü (2020) çalışmalarında, scrum yazılım geliştirme yöntemindeki başarı faktörlerine yer vermişlerdir [11]. Yıldız (2021) çalışmasında şelale ve scrum yöntemindeki tüm adımları karşılaştırmış ve karma yöntem hakkında bilgiler sunmuştur [12]. Yener, Baştürk ve Meçik (2019) çalışmalarında ise yazılım testindeki testlerin otomasyon ile nasıl verimlilik artışı elde edebileceklerini bir uygulama üzerinde göstermişlerdir. Bunun yanında otomasyon ve manuel (elle) yapılan testlerin karşılaştırmışlar ve hangi durumda hangi testin yapıldığını aktarmışlardır. Sonuçta test otomasyonu ile test yapma hızında artışlar sağlamışlardır [13].

Takgil ve Kara (2016), mobil uygulamaların testi için yaşanabilecek test ile ilgili zorluklara değinmişler ve bir otomasyon önerisinde bulunmuşlardır. Mobil uygulama testleri ile ilgili zorlukların başında test edilen cihaz ve üzerindeki mobil işletim sistemi ile ilgili zorluklardır şeklinde açıklamışlardır. Bu kapsamda yazılım testinde otomasyon ile insan hatalarının önlendiği, zaman kısıtının ortadan kalkabileceği, testlerin yeniden kullanılabilirliğinin artabileceğini ve herhangi bir zamanda testlerin çalıştırılabileceğini ifade etmişlerdir [14]. Khawaja ve Usman (2019) eserlerinde mobil uygulamaların artık web ve masaüstü uygulamalardan daha çok önemli bir noktaya geldiğini ve bu tür yazılımların kalitelerinin önemli olduğunu vurgulamışlardır. Bu kapsamda mobil uygulama kapsamında bir anket düzenlemişler ve bu doğrultuda hangi teknik ve araçların kullanıldığını araştırmışlardır. Buna göre fonksiyonel test en yüksek önem skorunu almıştır. Bunun ardından sırası ile performans testi, kullanılabilirlik testi, uyumluluk testi, güvenlik testi ve birlikte çalışabilirlik testi şeklinde sıralanmıştır [15].

Beşli ve Çavdar (2010) çalışmalarında yazılım projelerinde ve ardından özellikle veri ambarı projelerinde test prosedürleri hakkında bilgi sunmuşlarıdır [16]. Kılıç ve Öztürk (2010), entegrasyon testlerinde yaşanan problemleri entegrasyon zaman akışına göre üç alt grupta incelemişlerdir. Bunlar; kurulum aşaması, modüllerin uyumluluğun incelenmesi aşaması ve son olarak da entegre sistemde fonksiyonelite testi aşaması olarak belirtmişlerdir. Ardından entegrasyon stratejileri ve entegrasyon öncesinde test süreçlerinin önemi hakkında da bilgiler sunmuşlardır. Son olarak ise entegrasyon testlerini etkileyen diğer faktörleri aktarmışlardır [17]. Vural ve Sağıroğlu (2011) çalışmalarında güvenlik testleri, güvenlik testlerinin amaçları, güvenlik testlerinin sınıflandırılması, güvenlik testlerinin yapılması, bu konudaki standartlar ve kılavuzlar hakkında bilgi sunmuşlardır [18]. Uzun ve Koruyan (2019), eserlerinde test durum raporlarını incelemişlerdir. Buna göre hata tipine göre dağılımlar, hata şiddetine göre dağılımlar, hata önceliğine göre dağılımlar, hata durum dağılımı, test senaryosu ilerlemesi, hata ve senaryo sayısına göre analiz raporları şeklinde farklı bakış açılarına göre test durum raporlarını aktarmışlardır [19].

Yücalar ve Borandağ (2019) çalışmalarında Tümleşik Test Olgunluk Modeli (Test Maturity Model Integration-TMMI) hakkında bilgi sunmuşlardır. Bu kapsamda TMMI'ın yapısı, olgunluk düzeyleri ile ilgili bilgi verdikten sonra Tümleşik Yetenek Olgunluk Modeli (Capability Maturity Model Integration- CMMI) ile karşılaştırmışlardır. Sonuç olarak yazılım test süreçlerinin iyileştirilmesi kapsamında TMMI modelinin kullanılmasının önemini aktarmışlardır [20].

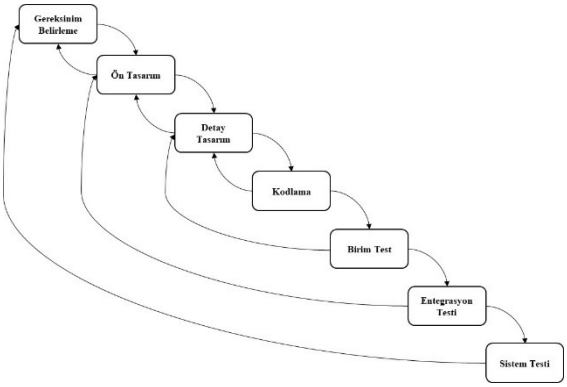
Procaccino, Verner ve Overmyer (2002) çalışmalarında yazılım geliştirmenin başarısı için yöneticilerin ve geliştiricilerin, müşterinin katılım sağlamasını dikkate almalarının önemli bir başarı etkeni olduğunu elde etmişlerdir. Bunun yanında müşterilerin gerçekçi beklentileri ve proje kapsamının

da net bir şekilde tanımlanmasının da yazılım geliştirmenin başarısı için önemli olduğunu açıklamışlardır [21]. Chow ve Cao (2008), çalışmalarında çevik yazılım geliştirmedeki başarısızlık faktörlerini dört ana grupta toplamışlardır. Bu dört ana grup, organizasyon, insan, süreç ve teknik şeklindedir. Başarı faktörlerine ise bunlara ilave olarak proje bacağınıda eklemişlerdir. Başarısızlık faktörleri altında 19, başarı fakörleri altında ise 36 kritere yer vermişlerdir [22]. Misra, Kumar ve Kumar (2009) çalışmalarında çevik yazılım geliştirme sürecinde 14 varsayımsal kritik başarı faktörlerinden dokuzunun başarı ile önemli ölçüde ilişkili olduğunu bulmuşlardır. Müşteri memnuniyeti, müşteri işbirliği, müşteri bağlılığı, karar zamanı, kurum kültürü, kontrol, kişisel özellikler, toplumsal kültür, eğitim ve öğrenim başarıda etkili olmuştur. Takım dağıtımı, ekip büyüklüğü, planlama, teknik yeterlilik ve iletişim ile olumsuzluğun başarı ile anlamlı bir ilişkisinin olduğu bulunmamıştır [23]. Nasir ve Sahibuddin (2011) çalışmalarında yazılım geliştirmedeki kritik başarı faktörlerini insan, süreç ve teknik konular şeklinde üç ana grupta ele almışlardır. Bu üç grupta yer alan 26 kritik başarı faktöründen en önemli ilk üç kritik başarı faktörü; gereksinimlerin ve spesifikasyonların açık olması, açık bir şekilde objektif/amaç/kapsam tanımı olması, gerçekçi bir zaman planı olması şeklinde sıralamışlardır [24]. Tsoy ve Staples (2020) çalışmalarında Chow ve Cao (2008)'un eserinde belirttiği kritik başarı faktörlerine ilaveler yaparak çevik analitik projelerinde araştırmalar yapmıştır. Sonuçta güçlü bir müşteri katılımı ve metodik bir proje sürecinin proje başarısı için önemli olduğunu görmüşlerdir [25].

2. *YÖNTEM (METHODS)*

Test işlemleri bize doğru ürüne/yazılıma sahip olunmasını sağlayan bir süreçtir. Test işlemi sonucunda, ilk başta ortaya konulan tasarıma göre yazılım, gerekli özellikleri yerine getirmekte midir? yoksa yazılımda hata var mıdır? şeklinde sorulara cevap aranmaktadır. Kalite bakış açısı ile test faaliyetleri kalite güvence amaçlı olarak kullanılmaktadır. Test etmenin bu kadar önemli olduğu bir durumda test faaliyetlerini ne zaman durdurmamız gerekmektedir? Bu konuda genel yaklaşım, test işlemlerinde bütçe, zaman ve kalite arasında bir dengenin kurulmasıdır. Test aktiviteleri test için ayrılmış olan bütçe, zaman veya test senaryolarından bir kısmı bitince de durdurulabilir. Bunun yanında yazılım, istenen güvenilirlik seviyesine gelince de başka bir yaklaşıma göre test aktivitelerini durdurmak gerekebilir. Bu alternatiflerden herhangi birisi aslında içinde bulunulan projeye göre farklılık arz edecek ve ilgili proje yöneticisi veya takım tarafından karar verilecektir.

Farklı yazılım geliştirme yaşam döngülerinde testler genel hatları ile benzerlik arz etmektedir. Şekil 1'de şelale yöntemindeki yazılım geliştirme sürecinin akışı ve bu yöntemde yer alan testler gösterilmektedir. Şelale yönteminde gereksinimler belirlendikten sonraki aşamada yazılımın ön tasarımı yapılmaktadır. Bu aşamada yazılımda yer alacak olan fonksiyonlar alt fonksiyonlara bölünmektedir. Dolayısı ile bu safha ile entegrasyon testi ilişkili bir şekilde yürümektedir. Çünkü bu bölünme sonraki aşamalarda entegrasyon testine tabi tutulacaktır. Ön tasarım sonrasında ise detay tasarım gelmektedir ve bu safhada ise bölünmüş olan her bir fonksiyonun altında birim seviyesine kadar inilerek tasarım yapılmaktadır. Bu safhanın karşılığı ise test aşamasında birim testlere denk gelmektedir. Bu safhada en alt seviyeye bölünen birimler, birim test aşamasında test edilmektedir. Sistem testi aşmasında ise gereksinimlerin tamamı ortaya koyulan sistem tasarımına göre doğru bir şekilde karşılanıp karşılanmadığı test edilmektedir.

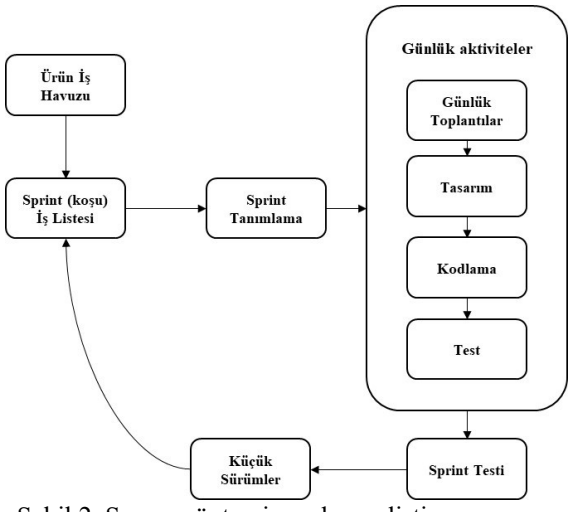


Şekil 1. Şelale yöntemi yazılım geliştirme yaşam döngüsü
(Waterfall Method Software Development Lifecycle)
(Kaynak: Jorgensen (2014)[26]

Jorgensen (2014) eserinde şekil 1'de yer alan şelale yönteminin faydalarını hiyerarşik yönetimler için uygun olduğunu, her bir safhanın ve sonunda çıkan ürünün tanımlı olduğunu ve detay tasarım safhasında her bir birimin belirlendiğini ve bunun sonucunda da farklı kişilerin bu birimde yazılım geliştirme yapabileceklerini ifade etmiştir. Bunun yanında şelale modelinin sorunlarını ise gereksinimlerin belirlenmesi ile test safhası arasında çok büyük zaman aralığı olduğunu ve ayrıca gereksinim safhasında çok iyi bir şekilde sistemin ihtiyaçlarının çıkarılması gerektiğini, burada yapılacak olan bir hatanın maliyetinin ilerleyen zamanlarda büyüyerek geleceğini aktarmıştır [26].

Boehm (1988), yazılım geliştirme ve iyileştirme için spiral model adlı çalışmasında spiral model ile ilgili bilgiler sunmuştur. Şelale yöntemindeki sorunların üstesinden gelebilmek için şelale yönteminin farklı şekilde uygulamaları olan artırımlı geliştirme, evrimsel geliştirme ve spiral geliştirme şeklinde farklı yaklaşımlar ortaya çıkmış ancak bu farklılıkların sonucunda da yazılıma her yeni eklenen fonksiyon için öncelikle regresyon testi ve ardından da yeni eklenen kısmın testinin yapılması gerekliliği doğmuştur. Aslında regresyon testi her hatanın düzeltilmesi veya her yeni fonksiyon testi sonrasında da yapılması gereken bir test olmasına karşın zaman ve bütçe kısıtlarından dolayı çoğu zaman atlanılmaktadır [27].

Şelale yöntemindeki sorunlar, ihtiyacın bir an önce karşılanması gereği ve farklı sebeplerden dolayı çevik yazılım geliştirme metotları ortaya çıkmıştır. Çevik yöntemler genel olarak; küçük fonksiyonlar şeklinde ve hızlı bir şekilde yazılım geliştirme faaliyetidir, bunun yanında gereksinimlerdeki değişim çok az olumsuz etkilemekte ve gereksinim belirleme ile test arasındaki zaman daralmıştır. Aşağıdan yukarıya bir geliştirme metodolojisi ve müşterinin sürekli işin içinde bulunduğu bir yöntemdir. Çevik yazılım geliştirme yöntemleri arasında XP (Extreme Programming), Test Güdümlü Geliştirme, Scrum gibi yöntemler sıralanabilir. Bu yöntemlerden en çok kullanılanı scrum'dır. Scrum yönteminin çalışması şekil 2'de görülmektedir.



Şekil 2. Scrum yöntemi yazılım geliştirme yaşam döngüsü

(Scrum Method Software Development Life Cycle)

(Kaynak: Jorgensen (2014)[26]

Buna göre scrum'da test faaliyetleri her bir kodlama sonrasında birim test şeklinde günlük yapılan kodlamanın ardından ve sprint sonrasında da kullanıma sunulacak olan kısımda entegrasyon testi yapılmaktadır. Ayrıca entegrasyon testinin yanısıra ürün sahibi tarafından da sistem testi şeklinde gereksinimleri karşılayıp karışılmadığı da test edilebilmektedir. Aslında scrum yöntemine, şelale yöntemindeki tüm fonksiyonların parçalı bir şekilde ele alınması şeklinde baktığımızda test faaliyetleri de şelale yöntemindeki yer alan büyük parçalar burada küçük parçalar halinde yapılmaktadır. Myers, Badgett ve Sandler (2012) yapmış oldukları Yazılım Test Sanatı adlı eserlerinde çevik test faaliyetlerinin ortak test formunda yürütüldüğünü ve bu kapsamda müşteri, yazılımcı, analist, testçi herkesin işin içinde olduğunu aktarmışlardır. Müşteriler kabul test kriterlerini belirttiklerini, programcıların test yapanlar ile test koşumlarını yaptıklarını ve fonksiyonları otomatik olarak test ettiklerini ve herkesin birbiri ile iletişim ve iş birliği içinde olduklarını aktarmışlardır [28].

Test aktivitelerinin diğer önemli bir kısmında fonksiyonel olmayan gereksinimlerin test edilmesidir. Yapılan proje veya sistemden farklı fonksiyonel olmayan gereksinimler olabilir. Bunlar; güvenlik, performans gereksinimleri, yer gereksinimleri, kullanılabilirlik, güvenilirlik gereksinimleri, sistem işletimi ile ilgili gereksinimler, kanuni olarak ortaya konulan gereksinimler şeklinde sıralanabilir. Bu gereksinimlerin sistem analizi aşamasında belirlenmesi ve sistemin ona göre tasarlanıp geliştirilmesi gerekmektedir. Test aşmasında ise bu fonksiyonel olmayan gereksinimlerin karşılanıp karşılanmadıklarının test edilmesi gerekmektedir. Bu tür gereksinimler tamamen sistemin altyapısını etkileyebileceği için önceden çok iyi belirlenmesi ve ona göre ilerlenmesinde fayda vardır. Sonradan küçük dokunuşlar ile bu tür gereksinimler ele alınamaz ve test edilemez durumda olabilirler. Bu kapsamda ele alınması gereken testler ise performans testi, stres testi, yük testi, güvenlik testi, ekran kullanım testi, hacim testi, uyumluluk testi şeklinde sıralanabilir.

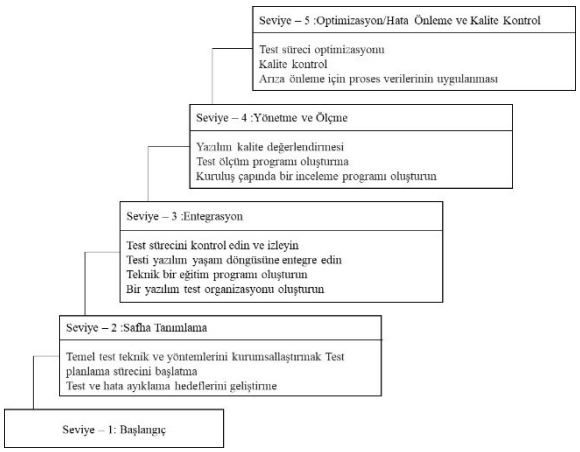
Fonksiyonel gereksinimlerin test edilebilmesi için özellikle test senaryolarının yazılması çok önemlidir. Test senaryoları her bir fonksiyonun test edilmesi açısından gerekli sayıda ve derinlikte ele alınmalıdır.

Bunun yanında testler sonucunda bulunan hatalarında iyi bir şekilde yönetilmesi önemlidir. Bulunan hata hangi test senaryosu ile elde edilmiştir. Bu test senaryosu hangi gereksinimi test etmek için oluşturulmuştur şeklinde bağlantıların kurulması da önemlidir. Bunun yanında bulunan hataların önceliği ve kritikliği de önemlidir. Eğer bulunan hatanın kritikliği, sistemi durduruyor, fonksiyon çalışmıyor veya veri kaybına neden oluyor ise yüksek kritiklikte olarak değerlendirilmelidir. Hata ara sıra oluyor ise

orta seviye, bazı durumlarda hata küçük bir etiket değişimi şeklinde olabilir ve bu durum az kritik seviye olarak seviyelendirilebilir. Bunun yanında bulgular bir şekilde öncelik sırasında ele alınmalıdır ki bir taraftan sistemi test edenler hatalar düzeltilirken test sürecine devam edebilmeleri gerekmektedir.

Myers, Badgett ve Sandler (2012) Yazılım Test Sanatı adlı eserlerinde kara kutu testinde, yazılımın içindeki yapı ve davranıştan ziyade yazılımın belirtilen spesifikasyonlara ne derecede uygun bir şekilde çalışıp çalışmadığına konsantre olunduğunu ifade etmişlerdir. Bunun yanında beyaz kutu testi ise yazılımın iç yapısına konsantre olmaktadır. Ayrıca test yapmak, yazılımdaki tüm hususların sorunsuz bir şekilde hatasız olacağını garanti etmemektedir. Yazılımın tamamını da test etmek maliyet etkin bir durum olmamaktadır. Bunun için test senaryolarının kapsamının çok iyi bir şekilde belirlenmesinde fayda vardır. Bu kapsamda test senaryoları birçok hatayı bulacak olasılıkta, sayıda ve içerikte olmalıdır. Genel olarak önerilen yapının test senaryolarının kara kutu testi metotları kullanılarak yapılması ve gerekmesi durumunda destekleyici test senaryolarının beyaz kutu testleri ile desteklenmesi gerekmektedir [28].

Burnstein (2003) Pratik Yazılım Testleri- Süreç Odaklı Bir Yaklaşım adlı çalışmalarında test olgunluk seviyesini yazılım olgunluk seviyesinde olduğu gibi 5 ana seviye üzerinden aktarmışlardır. Şekil 3'de görüldüğü üzere bu seviyeler; 1) Başlangıç, 2) Safha Tanımlama, 3) Entegrasyon, 4) Yönetme ve Ölçme, 5) Optimizasyon/Hata Önleme ve Kalite Kontrol şeklinde yer almaktadır [29].



Şekil 3. Test olgunluk seviyesi
(Test Maturity Level)
(Kaynak: Burnstein (2003)) [29]

Test olgunluk seviyesi test sürecinin nasıl iyileştirilebileceği ile ilgili olarak bize bilgi sunmaktadır. Bunun yanında bu olgunluk seviyesi, yazılımların test sürecinin ve test faaliyetlerine gereken önemin verilmesi sonucunda daha kaliteli bir yazılım ortaya konulmasına da fayda sağlayacaktır. Test olgunluk seviyesi, test sürecine odaklanmaktadır. Eğer test süreci olması gereken olgunluk seviyesinde olması durumunda zaten bu süreçte test edilen yazılımlar da (ürünler) kaliteli bir şekilde olması beklenmektedir.

Naik ve Tripathy (2008) yılında yapmış oldukları Yazılım Testi ve Kalite Güvence – Teori ve Pratik adlı eserlerinde güvenilirliğin kullanıcı odaklı kalite faktörü olduğunu ve tanımında yazılım sisteminin belirlenen bir zaman aralığında ve çevrede belirlenen görevi hatasız bir şekilde yapma olasılığıdır şeklinde belirtmişlerdir. Yazılım güvenilirliğini etkileyen faktörleri ise kodun karmaşıklığı ve büyüklüğü, geliştirme safhasının karakteristikleri, geliştiren personelin eğitimi, deneyimi ve işletim ortamının etkilemekte olduğunu açıklamışlardır. Sistem testleri yazılımdaki sorunları arındırmak için bir yöntem olmasından dolayı aynı zamanda yazlımın güvenilirliğinin sağlanmasında da çok etkin rol oynamaktadır [30].

Geliştirilen yazılımın testi için yazılacak olan test senaryoları yazılım geliştirme metodolojilerine göre hazırlanması farklılık gösterebilir. Eğer yazılım gereksinim ve tasarım safhasında kullanım senaryosu yazılırsa bu senaryoları çok kolay bir şekilde test senaryolarına döndürmek ve onun üzerinden testleri yapmak testin kalitesi açısından çok büyük önem arz etmektedir. Gereksinimlerin yazılması esnasında gereksinimlerin test edilebilir şekilde yazılması gereksinim kalitesi açısından önemlidir. Test senaryoları hazırlanırken testi yapacak olan kişi bu gereksinimlerden çok büyük oranda yararlanacaktır. Çevik yazılım geliştirime kapsamında oluşturulan kullanıcı hikayeleri de test senaryolarının oluşturulması esnasında veya kabul kriterleride testler esnasında bakılması gereken hususlardır.

Test süreci sonunda test edilen yazılımın gereksinimleri ne derecede karşıladığı ve bunun yanında müşteri tarafından istenilen ürünün ne derece isteklerini karşıladığı değerlendirilir. Bu aşamanın ilk kısmı proje ekibi tarafından yapılan testler aşamasında ortaya çıkmaktadır. Proje ekibinde yer alan iş analistleri sistemin tanımını yaptıkları anda aslında müşterinin isterlerini ürüne yansıtmaktadır ancak ne derece bilgi alabildilerse o derece yansıtabilmektedir. Proje ekibindeki test sürecinde de gereksinim toplama aşamasında ortaya konulan gereksinimlerin ne derecede doğru kodlandığı test edilmektedir. Müşteri tarafından yapılan testlerde ise müşteri kendi isteklerinin geliştirilmiş olan yazılımda yer alıp

almadığını test etmektedir ve test sürecine bakış açısı bu doğrultudadır.

3. *KRİTİK BAŞARI FAKTÖRLERİNİN ANALİZİ*
(ANALYSIS OF CRITICAL SUCCESS FACTORS)

Bu çalışma kapsamında tüm yazılım geliştirme süreci yerine farklı projelerdeki test süreci, bu süreçteki test aktiviteleri, test senaryoları, testler sonucunda bulunan bulguların nasıl ele alınacağı ile ilgili bilişim sistemi geliştirme projelerindeki test süreçlerinde dikkat edilmesi gereken hususlar kritik başarı faktörleri olarak Tablo 1'de ortaya konulmuştur. Değerlendirmeler mobil, web, kurumsal uygulamalar gibi farklı tipteki projelerin geliştirilmesindeki test süreçlerinin tamamı için yapılmıştır. Ayrıca, ortaya konulan kritik başarı faktörleri hem çevik hem de şelale yöntemi ile geliştirilen projeler için de uygulanabilecektir. Öncelik sıralaması kritik başarı faktörlerinden hangisinin en öncelikli olarak ele alınmasını ifade etmektedir. Bu sıralama orataya konulurken özellikle ilgili faktörün üç açıdan değerlendirilmesi sonucunda ortaya çıkmıştır. Bu faktörler proje (yazılım geliştirme projesi) zamanına, kalitesine ve bütçesine etkisi değerlendirilerek elde edilmiştir.

Tablo 1. Kritik başarı faktörleri ve öncelik sıralaması
(Critical success factors and order of priority)

No	Kritik Başarı Faktörü	Öncelik Sıralaması
1.	İnternet uygulamalarının test edilmesi	1
2.	Mobil uygulamaların test edilmesi	2
3.	Kurumsal uygulamaların test edilmesi	6
4.	Test sonuçları yönetimi	7
5.	Test kapsamı	8
6.	Test sürecinde zaman	9
7.	Test senaryolarının güncellenmesi	10
8.	Farklı duruma göre farklı test aktiviteleri	11
9.	Farklı yazılım geliştirme modellerinde test süreci	13
10.	Standardizasyon ve test sürecine etkisi	12
11.	Test yönetim aracının önemi	14
12.	Fonksiyonel olmayan gereksinimlerin test edilmesi	3
13.	Test yapan uzmanın ve test senaryolarının yazılım geliştirme sürecinin başında projeye dahil edilmesi	16
14.	İşletme'de diğer yazılım geliştirme süreçleri ile uyumlu tanımlanmış bir test sürecinin olması ve yönetim tarafından desteklenmesi	15
15.	Gereksinimlerin kalitesinin doğrulanması	4
16.	Test verisi	5

1. *İnternet uygulamalarının test edilmesi:* İnternete açık uygulamalar genel olarak tüm dünyaya açık olmasından dolayı böyle bir uygulama geliştirilmesi durumunda bu uygulamalar ile ilgili yazılım ve dolayısı ile sistem hızının test edilmesi en önemli hususdur. Bu tür uygulamalarda kullanıcılar, ilgili sayfanın gelmesi için çok fazla beklemek istemezler. İşlerinin yavaş olması durumunda hemen sistemden ayrılırlar veya bu yavaşlığı tekrar yaşamamak için bir daha bu sayfaya zorunlu olmadıkça geri gelmez. Ayrıca, internet sayfalarına çok fazla kullanıcının bağlanma durumuna göre yük testide dikkat edilmesi gereken diğer bir husustur. Bunun yanında bu tür uygulamaların, hem normal hem de kampanya gibi çok yoğun olabilecek dönemleri de dikkate alarak sistem testlerinin yapılmasında çok büyük fayda vardır. Sistemin önyüzü ve seçilen renkler bile kullanıcıları bir daha siteye tekrar getirecek şekilde olmalı ve kullanımı en deneyimsiz kullanıcının bile kullanabileceği yapıda olmalıdır. Sistemin 24 saat ayakta olacak şekilde kurgulanıp kurgulanmadığı da test edilmesi gerekmektedir. Hangi kullanıcının hangi zamanda sisteme bağlanacağı belli olmadığı zamanlarda sistemin sürekli ayakta olması önemlidir. Ayrıca, kullanıcıların farklı web tarayıcı kullanabilme durumlarına göre kullanımda yer alan farklı tarayıcılarda ve versiyonlarında sistemin nasıl davranış gösterdiği de test edilmesi gereken hususlar arasındadır. Uygulama üzerinde yer alan tüm bağlantıların da test edilmesi gerekmektedir.

2. *Mobil uygulamaların test edilmesi:* Mobil uygulamalar genel olarak işletme içinde veya işletme dışında kullanıcıların işlerini yaparken sabit bilgisayara ulaşma imkanlarının olmaması veya yaptıkları iş gereği bu şekilde çalışmalarından dolayı geliştirilmektedir. Dolayısı ile mobil uygulamalarda en önemli hususlardan bir tanesi mobil cihaz ile bu cihazın kullandığı ağ altyapısının çok iyi bir şekilde test edilmesi gerekmektedir. Özellikle düşük seviyeli hızlarda bile yazılımın çalışması önem arz etmektedir. Bunun yanında mobil uygulamalardaki en önemli hususlardan bir taneside güvenlik testleridir. Bunun yanında mobil cihazlarda çalışan işletim sistemi ve bunun yanında mobil cihazın ekranının büyüklüğü ve performansıda test edilmesi gereken hususlar arasında yer almaktadır.

3. *Kurumsal uygulamaların test edilmesi:* Kurmsal uygulamalar genel olarak kurum içinde iç müşteriler

için geliştirilmiş olan uygulamalardır. Bu uygulamalar için en önemli test aktivitesi ise test senaryolarının tam ve tüm gereksinimleri kapsayacak şekilde yazılması ve bunların test edilerek olumsuz sonuçlarının tamamlanması öenmlidir. Bu şekilde kullanıcı gereksinimleri yerine getirilmiş olacaktır. Bunun yanında bu madde sonrasında ele alınan tüm kritik test aktiviteleri bu tür uygulamalar içinde ele alınması gereken hususlardır.

4. *Test sonuçları yönetimi:* Test sonuçları resmi bir şekilde tutulmalıdır. Ayrıca testler sonucunda ortaya çıkan hatalar giderilmeden yazılım kullanıma açılmamalıdır. Eğer açılacaksa da testler sırasında testlerin sonuçlarının kritikliklerine göre sınıflandıracak bir yapı kurulması sistemin çalışmasını durduracak veya yanlış veri oluşmasına neden olacak test sonuçlarının tamamlanmadan sistemin işletime açılmaması gerekmektedir. Bunun yanında her bir test bulgusu hangi gereksinim ile ilişkilidir net bir şekilde geriye doğru ve ileriye doğru izlenebilirlik sağlanmalıdır. Bunun sonucunda da test ile ilgili farklı metrikler toplanabilir.

5. *Test kapsamı:* Tüm yazılım geliştirme projelerinde test süreci yazılımlardaki hata oranını azaltır. Bunları hiçbir zaman sıfıra indirmez. Hata oranlarını azaltmak amacı ile test senaryoları üzerinden ve çalışılan projeye göre en kritik olan gereksinimlerden başlayıp test etmek önemlidir. Bunun için bir öncelik matrisi yapılmalı ve buna göre hareket edilmelidir.

6. *Test sürecinde zaman:* Test sürecinde en büyük baskı zaman baskısıdır. Proje planı test sürecinide kapsayacak şekilde yapılır ancak test aktiviteleri yazılım geliştirme sürecinin son safhası olmasından dolayı projenin başında ki gereksinim belirleme ve sistem tasarımı ile ortasında yer alan kodlama aşamasında zaman tüketilir ve test aşamasına çok az zaman kalır. Bu durum, test yapan personelin üzerinde proje yönetimi veya üst yönetim tarafından bir baskı unsuru olarak yer alır. Test sürecinin bu kısıtlı zamanda ve zaman baskısı altına yapılması durumunda elde edilen yazılımda sorunlar daha çok olacaktır. Bu durum işletim aşamasında ortaya çıkacak mutsuz müşteri ve projenin bütçesine olumsuz yönde etki edecektir. Bunun için test sürecine gelmeden önce zamanı iyi bir şekilde yönetmeli ve test sürecinin planlandığı şekli ile süresi içinde yapılması sağlanmalıdır. Bunun yanında belki de test süreci için yazılım geliştirmenin sonu beklenmemeli projenin öncesinde test senaryoları hazırlanmalı ve bununla ilgili olarak bu senaryoların koşulacağı zaman belirlenerek proje yönetimine bilgi verilmelidir.

7. *Test senaryolarının güncellenmesi:* Bazı durumlarda test senaryolarının yenilenmediği ve bunun sonucunda da yazılımda yer alan sorunların bu senaryoların koşumu sonucunda bulunamadığı gözlemlenmiştir. Bunun önüne geçmek için test senaryoları belirli aralıklar ile yazılıma ilave modüller veya yetenekler eklendikçe güncellenmeli ve bu yeni kısımlar içinde yeni test senaryoları yazılmalıdır.

8. *Farklı duruma göre farklı test aktiviteleri:* Yazılım çeşidine, yazılımın çalışacağı ortama, yazılımı kullanacak olan müşterilerin durumlarına, yazılım geliştirme yöntemlerine veya farklı durumlara göre test aktiviteleri ve test tipleri farklılık arz etmektedir. Tek bir test çeşidi ve yöntemi bu farklılıkları ele alması beklenmemelidir. Onun için ihtiyaca göre farklı testler ele alınmalı ve yapılmalıdır. Ayrıca farklı projelerden elde edilen test ile ilgili deneyimler bir sonraki projelere yansıtılmalıdır ki hem farkındalık yaratılsın hem de deneyimler sonucunda daha hızlı çözümler üretilebilsin. Projenin bir tanesi mobil bir taneside web projesi olması durumunda test gereksinimleri her iki projede de farklılık arz etmektedir. Tabiki fonksiyonel test kısımları gereksinimlere göre benzerlik arz edecektir. Bu kapsamdaki testler her iki projede de aynı olacaktır. Ancak projelerin çalışma ortamları gibi fonksiyonel olmayan gereksinimlerin test edilmesi projelerden projelere farklılık arz edecektir. Dolayısı ile bu farklılıklar yazılımın gereksinimlerinin belirlenmesi aşamasında görülmesi ve ona göre test planı çıkarılması gerekmektedir.

9. *Farklı yazılım geliştirme modellerinde test süreci:* Yazılım geliştirme projelerininde, özellikle çevik yazılım geliştirmelerde amaç kullanıcıya küçük fonksiyonlar şeklinde yazılımları geliştirerek hem geliştirme hatalarını azaltmak hem de kullanıcıya bir an önce yazılımın ilk kısımlarını kullanmasını sağlamaktır. Bunun yanında V model veya şelale yönteminde de yazılımlar belirli fonksiyonlar üzerine eklenerek geliştirilirler ve aralarda bazı fonksiyonlar bitince test yapan personel test çalışmalarına başlamaktadır. Bu durum özellikle yeni fonksiyon veya özellik eklenince yazılımın altyapısı düzgün bir şekilde kurulmadı ise veya deneyimsiz yazılımcılar ile çalışılması durumunda daha önce test edilen veya kullanıcıya açılmış olan kısımlarda bile sorunlar çıkabilmektedir. Eğer test yapan personel bu kısımları tekrar test etmez ise eskiden test edilmiş ve doğrulanmış olan kısımlarda da sorunlar çıkmaya başlayacaktır. Bu şekilde ki geliştirmelerde özellikle bu hususlara dikkat edilmeli ve regresyon testleri yapılmalıdır.

10. *Standardizasyon ve test sürecine etkisi:* Standartlaşma yazılım geliştirme projelerinde çok

büyük öneme sahiptir. Bu kapsamda kodlama, gereksinim belirleme ve sistem tasarımı, test aşamalarındaki kontrol listeleri, standartlar büyük öneme sahiptir. Her yazılımcı kendi bilgi birikimi ve çalışma şekline göre kodlama yapması durumunda ortaya çıkan ürünün hem testi hem de inceleme ve değerlendirmesi zor olacaktır. Bunun için standartlaşmaya ayrı bir önem verilmesi projelerin başarısı için önemli bir husustur.

11. *Test yönetim aracının önemi:* Test aktivitelerinin tamamını yazılım geliştirme yaşam döngüsündeki diğer safhalardaki aktiviteler ile ilişkili bir şekilde takip edilmesini sağlayacak bir aracın kullanılması projenin başarısı için ve hatta buradaki verilerden dersler çıkarmak amacı ile de önemlidir. Dolayısı ile işletmenin yazılım geliştirme sürecinde test aktivitelerinde unutulmaması ve bu sürecin bir bütün olarak ele alınıp ona göre çözüm sağlayacak olan bir aracın kullanılması önemlidir. Eğer test aktivitelerinin yönetimi ayrı bir yerde el ile veya ilişkili bir şekilde tutulmaması durumunda bazen karışıklıklar yaşanabilecek ve aralarda yapılmayan ve unutulan testler sonucunda bulunan hataların düzeltilmeden işletime verilen yazılımlar olabilecektir. Bu durum, karşımıza istemediğimiz sonuçlar doğurabilecektir.

12. *Fonksiyonel olmayan gereksinimlerin test edilmesi:* Test sürecinde bir başka önemli husus fonksiyonel olmayan gereksinim testlerinin projenin içeriğine göre planlanması ve yapılmasıdır. Bu kapsamda özellikle sistemin performansı son kullanıcı için çok önemli bir husustur. Eğer proje webe açık bir uygulama ise son kullanıcılara bir saniye bile geç şekilde yazılımın cevap vermesi durumunda eğer kullanıcı müşteri olacaksa belki de siteden ayrılacak ve müşteri olmaktan vazgeçecektir. Eğer sistem askeri veya gizlilik anlamında bir içeriğe sahip ise bu durumda da güvenlik anlamında fonksiyonel olmayan testlere önem verilmesi ve bu testlerin en küçük güvenlik açığı oluşturmayacak şekilde yapılması gerekmektedir. Bunların yanında bazı durumlarda da sistemin kullanılabilirliği çok önemlidir. Kullanılabilirlik, eğer kullanıcılarınız çok farklı seviyede ise bu konu daha da önemli hale gelmektedir. Bu durumda en kolay kullanım yöntemi farklı çalışmalar sonucunda bulunmalı ve bu kapsamda kullanılabilirlik testleri yapılmalıdır. Kullanımı zor, ekrana kullanıcı girdiğinde ne yapacağını tam olarak anlayamıyorsa sistemin bu şekilde başarı sağlaması çok zordur. Bu durum hem müşteri kaybına hem de farklı kayıplara sebep olabilecek ve projenin başarısını etkileyecek bir duruma gelebilecektir.

13. *Test yapan uzmanın ve test senaryolarının yazılım geliştirme sürecinin başında projeye dahil edilmesi:* Hem çevik yazılım geliştirme hem de şelale yönteminde veya firmanın kendine özgü karma bir metodolojisi olsa bile testi yapacak olan personelin projenin başından itibaren iş analisti ile beraber çalışmaya başlaması ve ilgili gereksinimler belirlenirken test yapacak olan uzman veya mühendisin de test için gerekli olan test senaryoları ile ilgili çalışmaları yapması, proje sonucunda ortaya çıkan yazılımın hatasız bir şekilde çıkması için çok önemlidir. Test uzmanının proje kapsamında bakış açısı ortaya konulmuş olan bir yazılımı nasıl bozabilirim veya hataya düşürebilirim, iş analistinin bakış açısı ile bu yazılımın temellerini nasıl sağlam atabilirim, yazılım uzmanının bakış açısı ise bu yazılımı nasıl en doğru bir şekilde inşa edebilirim şeklindedir. Dolayısı ile test uzmanının bakış açısından ele alındığında yazılımın, işletim aşamasında hata vermemesi için en başarılı bir şekilde testi yapılmalıdır.

14. *İşletme'de diğer yazılım geliştirme süreçleri ile uyumlu tanımlanmış bir test sürecinin olması ve yönetim tarafından desteklenmesi:* Eğer test süreci tanımlı değil ve test işlemleri yazılım geliştirme sürecindeki kişilerin inisiyatifi ile oluyor ise ve çok basit bir şekilde yapılıyorsa projenin uygulamaya alma esnasında çok büyük sıkıntıların olması kaçınılmazdır. İşletmedeki test süreci, olgunluk seviyesine göre incelenmeli ve bununla ilgili iyileştirme çalışmaları başlatılarak en iyi duruma nasıl getirileceği üzerinde bir proje çalışması başlatılmalıdır.

15. *Gereksinimlerin kalitesinin doğrulanması:* Yazılımın temelini oluşturan gereksinimlerin kaliteli bir şekilde belirlenip belirlenmediği doğrulama aşamasında ele alınmalıdır. Bu kapsamda gereksinimler doğru, tam, uyumlu, test edilebilir, gerekli, uygulanabilir, net, izlenebilir ve önceliklendirilmiş olması önemlidir. Çünkü, bu gereksinimlerin bu kriterler ışığında ele alındığı doğrulanır ise test aşamasında daha düzgün ve daha iyi test senaryoları ele alınabilir ve testler yapılır. Aksi takdirde testlerin yapılması ve yazılımda yer alan hataların ortaya çıkarılması zor olacaktır. Örneğin, bu kalite gereksinimlerinden en önemlisi gereksinimlerin test edilebilir bir şekilde yazılmasıdır. Bunun yanında eğer gereksinimler kritikliklerine göre veya bir şekilde bir önceliğe göre sıralanır ve buna göre geliştirme olur ise test planlamasının yapılması da bu denli kolay olacaktır.

16. *Test verisi:* Test aşamasında test edilecek olan yazılımdaki tüm fonksiyonları kapsayacak şekilde test verisi oluşturmak çok önemlidir. Test verisi hazırlamak ayrı bir uzmanlık gerektirmektedir. Bazı durumlarda işletim aşamasındaki veriden de yararlanılabilir. Test verisi oluşturma aşamasında ise

hazır test verisi oluşturma yazılımları da bir alternatif olarak kullanılabilir. Test yapmaya başlamadan önce gerekmesi durumunda test verisinin yedeği alınmalı ve daha sonraki durumlarda test verisi tekrar tekrar kullanılması gerekmesi durumunda bu veriden yararlanılabilir.

4. *SONUÇ VE ÖNERİLER* *(CONCLUSION AND RECOMMENDATIONS)*

Yazılım kalitesi hem son kullanıcılar hem de projede çalışanlar için son derece önemlidir. Yazılımda kaliteyi sağlayan en büyük yöntemlerden birisi de yazılım geliştirme sürecinde, test çalışmalarının geliştirilen yazılımın çeşidene göre profesyonel bir şekilde yapılması gerekmektedir. Bunun yanında projenin test çalışmalarında yaşanmışlıklardan örnek alarak aynı sorunları tekrar tekrar yaşamamak gerekmektedir. Bunun için özellikle yazılımın başarılı olabilmesi için test safhasına ayrı bir önem verilmeli ve projenin tamamlanması için zaman kalmadı şeklinde yaklaşımlar sergilemeden yazılım için gerekli tüm testler yapılmaldır. Test sürecinde yazılımda yer alan tüm fonksiyonlar gerekli tüm testlerden proje bütçesinin elverdiği oranda test edilmeli ve bunun sonucunda da müşteriye, hatasız bir ürün elde etmesi sağlanmalıdır. Ayrıca bunun yanında proje geliştiricileri de işletim aşamasından gelecek olan sorunlar ile uğraşmak zorunda kalmayacak ve sonuçta bu sorunlar ile ilgili çalışanları başka projeler de çalıştırarak kazanç elde etmiş olacaklardır. Bu çalışma kapsamında ortaya konulan kritik başarı faktörleri, öğrenilen dersler, çalışmalar ve incelemelerden ortaya çıkmış ve başka test uzmanları veya yazılım geliştirme projesinde çalışan tüm farklı roldeki çalışanlar için yol gösterici niteliğindedir. Bu çalışma sonrasında spesifik olarak her bir test alanı ile ilgili kritik başarı veya başarısızlıkları önlemek amacı ile nelere dikkat edilmelidir şeklinde farklı çalışmalar yapılabilir.

KAYNAKLAR *(REFERENCES)*

[1] V. R. Basili ve R. W. Selby, "Comparing the Effectiveness of Software Testing Strategies" *IEEE Transactions on Software Engineering,* cilt 13, no. 12, pp. 1278-1296, 1987.

[2] D. Gelperin ve B. Hetze, "The Growth of Software Testing" *Communications of the ACM,* cilt 51, no. 6, 1988.

[3] J. A. Whittaker, "What is software testing? And why is it so hard? " *IEEE Software,* cilt 17, no. 1, pp. 70-79, 2000.

[4] G. Tassey, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, RTI for National Institute of Standards and Technology, 2002.

[5] E. Dustin, *Effective Software Testing – 50 Specific Ways to Improve Your Testing*, Pearson, 2003.

[6] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams" *Future of Software Engineering (FOSE '07)*, 2007.

[7] A. Orso ve G. Rothermel, "Software Testing: A Research Travelogue (2000–2014)" *FOSE 2014: Future of Software Engineering Proceedings*, 2014.

[8] A. Baran, F. Akar ve F. Aslay, "Çevik Yazılım Geliştirme Yöntemlerinde Etkili Test Araçları" *International Science and Technology Conference*, Vienna-Austria, 2016.

[9] A. Baran, F. Aslay ve F. Akar, "Geleneksel Yazılım Geliştirme Yöntemlerinde Test Problemleri" *3. Uluslararası Yönetim Bilişim Sistemleri Konferansı*, İzmir, 2016.

[10] A. K. Arumugam, "Software Testing Techniques & New Trends" *International Journal of Engineering Research & Technology (IJERT),* cilt 8, no. 12, 2019.

[11] A. Aydıner, M. Esen ve E. Özlü, "Türkiye'de Çevik Yazılım Geliştirme Süreçlerinde Scrum Yöntemini Uygulayan İşletmelerin Başarı Faktörleri", *Bilişim Teknolojileri Dergisi*, c. 13, sayı. 4, pp. 463-477, 2020.

[12] D. Yıldız, "Bilişim Sistemi Yazılım Geliştirme Sürecinde Kullanılan Şelale ve Scrum Yöntemlerinin Karşılaştırılması Ve Karma Yöntemin İncelenmesi", *Yönetim Bilişim Sistemleri Dergisi*, c. 7, sayı. 1, pp. 24-43, 2021.

[13] H. A. Yener, F. Baştürk ve B. Meçik, "Yazılım Geliştirme ve Test Otomasyon ile Verimlilik Artışı: General Mobile", *Karamanoğlu Mehmetbey Üniversitesi Mühendislik ve Doğa Bilimleri Dergisi*, c. 1, sayı. 1, pp. 172-196, 2020.

[14] B. Takgil ve R. Kara, "Android Mobil Uygulamalar için Yazılım Testi" *El-Cezerî Fen ve Mühendislik Dergisi,* cilt 3, no. 2, pp. 324-328, 2016.

[15] A. S. Khawaja ve A. Usman, "Mobile Application testing tools and their challenges: A comparative study" *2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2019.

[16] O. Beşli ve İ. H. Çavdar, "Veri ambarı Yazılım Geliştirme Sürecinde Test" *Akademik Bilişim'10 - XII.*

Akademik Bilişim Konferansı Bildirileri, Muğla, 2010.

[17] E. Kılıç ve S. Öztürk, "Büyük Ölçekli Yazılım Projelerinde Entegrasyon Testleri" *2. Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu*, İstanbul, 2010.

[18] Y. Vural ve Ş. Sağıroğlu, "Kurumsal Bilgi Güvenliğinde Güvenlik Testleri Ve Öneriler" *Gazi Üniv. Müh. Mim. Fak. Der.,* cilt 26, no. 1, pp. 89-103, 2011.

[19] B. Uzun ve K. Koruyan, "Yazılım Test Sürecinde Durum Raporlamasına Genel Bakış ve Yaklaşımlar" *Yönetim Bilişim Sistemleri Dergisi,* cilt 5, no. 1, pp. 52-63, 2019.

[20] F. Yücalar ve E. Borandağ, "Yazılım Projelerinde Kalitenin Arttırılması: TMMi" *AURUM Mühendislik Sistemleri Ve Mimarlık Dergisi,* cilt 3, no. 2, 2019.

[21] J. D. Procaccino, J. M. Verner, and S. P. Overmyer, "Case Study: Factors for Early Prediction of Software Success & Failure", *Information and software technology*, c. 44, sayı. 1, pp. 53-62, 2002.

[22] T. Chow, D. Cao, "A survey study of critical success factors in agile software projects", *The Journal of Systems and Software,* sayı. 81, pp. 961-971, 2008.

[23] S. C. Misra, V. Kumar, U. Kumar, "Identifying some important success factors in adopting agile software development practices", *The Journal of Systems and Software,* sayı. 82, pp. 1869-1890, 2009.

[24] M. H. N. Nasir, S. Sahibuddin, "Critical success factors for software projects: A comparative study", *Scientific Research and Essays,* c. 6, sayı. 10, pp. 2174-2186, 2011.

[25] M. Tsoy, D. S. Staples, "What Are the Critical Success Factors for Agile Analytics Projects?", *Information Systems Management,* sayı. September, pp. 1-18, 2020.

[26] P. C. Jorgensen, *Software Testing A Craftsman's Approach*, Fourth Ed.., CRC Press, 2014.

[27] B. W. oehm, "A spiral model for software development nd enhancement" *IEEE Computer,* cilt 21, no. 6, pp. 61-72, 1988.

[28] G. J. Myrs, T. Badgett ve C. Sandler, *The Art of Software Tesing*, Third Ed., John Wiley and Sons, 2012.

[29] I. Burstein, *Practical Software Testing A Process Oriented Approach*, Springer, New York, 2003.

[30] K. Naik ve P. Tripathy, *Software Testing and Quality Assurance - Theory and Practice*, New Jersey: Wiley, 2008.