REVİZYON TABLOSU

REVİZYON NO	REVİZYON GEREKÇESİ	TARİH
01	2022 versiyon değişikliği	01.10.2024

1. AMAÇ

Aksaray Üniversitesi'nin bünyesinde geliştirilen ya da satın alınan yazılımların testi ile ilgili genel işlemlerin prosedür dahilinde yapılmasını sağlamak.

2. KAPSAM

Kurumumuz genelinde satın alınan ya da geliştirilen tüm yazılımlar.

3. GİRİŞ

Geliştirilen yazılımlarda, kaynağı ve sebebi değişmekle birlikte çeşitli hataların olması kaçınılmazdır. Çalışılan uygulama alanının kritikliğine göre bu hataların doğuracağı sonuçların biçimi değişebilmektedir. Bir finans uygulamasında yapılan küçük bir hata çok büyük miktarlarda para kaybına yol açabilecektir. Bu nedenle profesyonel kullanımı planlanan tüm yazılımların içerisindeki hataların bulunması ve düzeltilmesi gereklidir. Bu çalışmalar için geliştirme işçiliğinin en az üçte biri kadar işgücü ayrılması önerilmektedir. Yazılım test çalışmaları, geliştirilen ve/veya geliştirilmekte olan yazılımlar içerisindeki hataların bulunup, düzeltilmesi, yazılımların hata içermediğinin garantilenmesi ve yazılımların doğru çalıştığının gösterimi amacı ile gerçekleştirilen faaliyetlerdir. Bu çalışmalar, geliştirme sürecinde gereksinimlerin belirlenmesi aşamasında başlayıp, yazılımların/ sistemlerin kurumumuza uygulanmasına kadar devam eden bir süreçte belirli disiplinlere uygun olarak çalışılmasını gerektirmektedir. Yazılım test faaliyetleri sadece geliştirilen kod parçaları üzerindeki hataların bulunması faaliyetlerini değil, geliştirme çalışmaları sırasında hata oluşmasını önleyecek yöntem ve yaklaşımların belirlenmesi faaliyetlerini de içerir. Geliştirme sürecinin başlangıç aşamalarında tespit edilen bir hatanın maliyeti ile bir sistem üzerinde tespit edilen bir hatanın maliyeti arasında ciddi farklar olabilmektedir. Bu nedenle olası hataların geliştirme aşamasının mümkün olduğunca erken safhalarında bulunması için yapılan gözden geçirme ve tasarım doğrulama faaliyetleri gibi çalışmalar da doğrulama/ geçerlilere sürecinin bir parçasıdır.

Bu dokümanda, ağırlıklı olarak kurumumuzda de kullanılan yazılım ve donanımların yapılan doğrulama faaliyetleri hakkında bilgi verilmektedir. Projeler içerisinde yer alan yazılımların doğrulanması için uygulanan süreçler ve bu süreçlerin nasıl uygulandığı anlatılmaktadır. Doğrulama süreçleri içerisindeki aşamalar, çıktılar, roller ve kullanılan araçlara değinilmektedir. Hata tespit ve düzeltme faaliyetlerinin yanı sıra hata önleme amacı ile yapılan faaliyetler hakkında da bilgi verilmektedir. Sonuç bölümünden önce hata önleyici faaliyetlere kısaca değinilecektir.

4. GÜVELİ YAZILIM GELİŞTİRME

Yazılım Geliştirme Süreçleri

Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar sürekli bir döngü biçiminde ele alınmalıdır. ASÜ bünyesinde güvenli yazılım geliştirme süreçleri aşağıdaki adımlarda ele alınmaktadır.

4.1. ANALİZ

Bir problemin çözümü olarak nitelediğimiz yazılımların ne yapacağını ve nasıl yapacağını analiz aşamasında belirlenir. Bu analiz sonucu öncelikle yazılımdan ne istendiğinin doğru bir biçimde tanımlanır ve kapsam belirlenir. Analiz aşaması personel, donanım ve sistem gereksinimlerinin belirlenmesi, sistemin fizibilite çalışmasının yapılması, kullanıcıların gereksinimlerinin analizi, sistemin ne yapıp ne yapmayacağının kısıtlamalar göz önüne alınarak belirlenmesi, bu bilginin kullanıcılar tarafından doğrulanması ve proje planı oluşturulması adımlarından oluşur.

4.2. TASARIM

Analiz aşaması sonucunda belirlenen gereksinimlere yanıt verecek yazılımın temel yapısının oluşturulduğu aşamadır. Yazılım tasarımı, bir bileşen veya sistemin nasıl gerçekleştirileceğini belirlemek için kullanılan teknikler, stratejiler, gösterimler ve teknikleri içermelidir. Bu aşama yazılım bileşenleri arasındaki içsel ara yüzler, mimari tasarım, veri tasarımı, kullanıcı ara yüzü tasarımı, tasarım araçları ve tasarımın değerlendirilmesi alt süreçlerini detaylı olarak tasarlanır ve dokümante edilir. Tasarım aşaması, yazılımın hem kullanıcı ara yüzünü hem de programın omurgasını ortaya koymalıdır. Yapılacak tasarım, yazılımın işlevsel gereksinimlere uygun olmasının yanı sıra kaynaklar, performans ve güvenlik gibi kavramları da göz önüne alınarak gerçekleştirilir. Bu bilgiler ışığında proje planı oluşturulur.

4.3. KODLAMA

Kodlama aşaması, tasarım sürecinde ortaya konan veriler doğrultusunda yazılımın gerçekleştirilmesi aşamasıdır. Bu süreç programlama çalışmalarının yanı sıra yazılımın geliştirilmesi ve kullanıcıya ulaştırılması sürecindeki bütün çalışmaları kapsar. Tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışması olarak da nitelendirilebilir. Yazılım geliştirme ortamı, programlama dili, veri tabanı yönetim sistemi, yazılım geliştirme araçları seçimi kodlama aşamasında mutlaka şeçilir ve tüm sistem seçilen bu kodlama mantığı üzerinde gerçekleştirilir. (Örnek Visual Studio Ortamında. Net Framework 4.5 ile Devexpress 15.1 tool kullanılarak yazılacak projenin tüm raporları ve çıktıları FastReport uygulaması ile geliştirilecektir.)

4.4. TEST

Test aşaması, yazılım kodlanması sürecinin ardından gerçekleştirilen sınama ve doğrulama aşamasıdır. Tüm yazılımlar için mutlaka yazılım test ortamı oluşturulur. Elde edilen uygulama yazılımının hem belirlenen gereksinimleri sağlayıp sağlamadığı hem de gerçekleştirimin beklentilere uygun olup olmadığını kontrol etmek için statik ve dinamik sınama teknikleri uygulanır. Madde 8 de detaylı açıklanan Yazılım test prosedürü içerisindeki konular dikkate alınarak test kriterleri ve metotları uygulanır. Statik teknikler, yazılımın tüm yaşam döngüsü boyunca elde edilen gösterimlerin analizi ve kontrolü, dinamik teknikler sadece gerçekleştirilmiş sistemi içermelidir. Yazılım üretiminde ilk testler genelde geliştirme sürecinde programcı tarafından yapılır.

Bununla birlikte, asıl hata ayıklama ve geribildirim hizmeti test ekipleri tarafından yapılır. Testler ve geribildirim yazılımdaki tüm değişiklik ve her yeni geliştirme sürecinde devam eder. Yazılımı kullandığı sürece devam eder. Test sürecinde en faydalı geribildirimler son kullanıcı test gruplarından gelir.

4.5. BAKIM

Yazılımın tamamlanmasından sonra hata giderme ve yeni eklentiler yapma aşamasıdır. Yazılımın kullanıma başlanmasından sonra yazılımın desteklenmesi sürecini kapsar.
Yazılımın eksiklerinin giderilmesi, iyileştirilmesi gibi alt aşamaları içeren aşamadır.

5. GÜVENLİ YAZILIM GELİŞTİRME

Yazılım güvenliği kavramı "güvenilir bilişim" (trusted computing) kavramı ile içi içe değerlendirilmelidir. "Trusted Computing Group" tarafından konmuş olan güvenilir bilişim kavramı gizlilik, bütünlük, erişebilirlik ve kurtarılabilirlik olmak üzere dört temel kavram üzerinde bütünleşir.

Güvenli yazılım geliştirme süreçlerinde ayrıca değişiklik ve konfügrasyon yönetimi, geliştirme, test ve üretim ortamı ayrışımı, geliştirme ortamında gerçek verilerin kullanılmaması, üretim ortamına almadan önce kod incelemesi, güvenli programlama teknikleri kullanımı, uygulama güvenlik duvarı kullanımı ya da kaynak kod inceleme hizmeti alınması gibi çalışmaların yapılması da güvenliğe ayrıca katkı sağlar.

Güvenli yazılım geliştirme sürecinde ele alınması gereken temel olarak dokuz ana güvenlik konusu vardır:

5.1 Girdi Geçerleme (Input Validation):

Günümüzde bilinen tehditlerin çoğu kötü niyetli girdi ile başlamaktadır. Bununla birlikte; basit girdi geçerleme yöntemleri ile büyük güvenlik tehditlerinin önlenmesi mümkündür.

Girdi geçerleme yöntemlerini "beyaz kutu" ve "kara kutu" olmak üzere ikiye ayırmak mümkündür. Beyaz kutu yönteminde bilinen bir şablon girdi olarak kullanılmakta, bu şablonun dışındaki tüm girdiler kötü niyetli olarak kabul edilmektedir. Şablonun kontrolü çok kolay olduğundan bu yöntem oldukça etkili bir yöntemdir. Kara kutu yöntemi ise daha az etkili olmasına rağmen daha çok tercih edilen bir yöntemdir. Bu yöntemde kullanılan belirli bir şablon yoktur, sadece bilinen saldırıların bir listesi mevcuttur. Eğer girdi bilinen bir saldırıya benziyor ise o zaman girdi reddedilecek, onun dışındaki tüm girdiler ise kabul edilecektir. Dolayısıyla veri yapıları, mümkün olduğunca belli bir şablona uygun tasarlanarak geçerleme daha güçlü kılınmalıdır.

İstemci-sunucu uygulamalarında geçerleme hem istemci hem de sunucu tarafında yapılmalıdır. Bununla birlikte; bir saldırgan istemci tarafındaki geçerleme kontrolünü kolay aşabileceğinden istemci tarafındaki geçerleme hiçbir zaman yeterli bir güvenlik önlemi olarak ele alınmamalıdır. Bunun yerine daha çok sunucu tarafında geçerleme kontrolü yapılarak güvenlik seviyesi artırılacaktır. Kısaca güvenilir olmayan bir kaynaktan (örneğin kullanıcıdan) gelen veri mutlaka kontrol edilmelidir.

5.2 Kimlik Doğrulama (Authentication):

Kimlik doğrulama, varlıkların (kullanıcı, cihaz veya bir uygulama) kimlik kontrolünden geçmesi işlemidir.

Yazılımda sadece kullanıcı adı ve şifre kullanması şeklinde zayıf doğrulama yöntemleri yanı sıra program amacı doğrultusunda mümkünse yazılım "domain" yapısı üzerinde, "Active Directory","SSO" kullanılarak doğrulanmalıdır. Domain dışında ise kimlik yönetimine ilişkin veritabanı uygulanmalıdır. Daha güçlü doğrulama yöntemi olarak da akıllı kartlar kullanılmalıdır.

Yazılım yanlış kimlik doğrulama denemelerine karşın mutlaka deneme yapılan hesabı blok etme özelliğine sahip olmalıdır.

5.3 Yetkilendirme (Authorization):

Kullanıcıların tanımlanması aşaması olan kimlik doğrulamadan sonra kullanıcının kimliği doğrultusunda erişim haklarının belirlendiği ve kontrolünün gerçekleştiği aşama yetkilendirmedir. Her uygulama ekranında yetkilerle işlem mutlaka yapılmalıdır.

5.4 Konfigürasyon Yönetimi (Configuration Management):

Konfigürasyon, uygulama ile ilgili hassas bilgileri içermektedir. Konfigürasyon dosyaları mutlaka sunucularda saklanmalı, konfigürasyon dosyaları hassas bilgi olarak nitelendirilmeli ve bu dosyalara erişim kayıt altında tutulmalıdır. Yazılım bir ekip tarafından geliştiriliyor ise mutlaka TFS (Team Foundation Server) ya da Agile Git gibi geliştirme ortamları mutlaka kullanılmalıdır.

5.5 Hassas Bilgi (Sensitive Information):

Hassas bilginin ne olduğunun belirlenebilmesi için uygulamanın ve işin bir arada ele alınması gerekir. Uygulama geliştirici işin niteliğini tam olarak bilemediğinden, diğer yandan işin sahibi de uygulamanın teknik altyapısı hakkında sınırlı bilgiye sahip olacağından bu iki taraf tek başlarına hassas bilgi için yeterli tanımlama yapamayacaklardır. İki tarafın bir araya gelmesiyle hassas bilgileri içeren bir liste oluşturulmalı ve bu listeyi koruyacak bir politika oluşturulmalıdır. Bu politika mutlaka proje planında belirtilmelidir.

5.6 Kriptografi (Cryptograhy):

Veriyi korumanın yollarından biri de şifrelemedir. Hassas bilgiler bilinen ve test edilmiş şifreleme yöntemleri ile saklanmalıdır. Ayrıca daha önce kırılması uzun zaman alan algoritmalar günümüzde daha kısa zamanda çözülebilmektedir. Dolayısıyla uygulama içindeki algoritmalar zamanla gözden geçirilmeli ve güncellenmelidir. (Örnek olarak client uygulamalarda yüklenen dll dosyalarının içerikleri şifrelenebilir.)

5.7 Parametre Manipülasyonu (Parameter Manipulations):

Dağıtık algoritmalar modüller arasında parametre gönderirler. Eğer bu parametreler arada değiştirilirse, saldırı gerçekleştirilmiş olur. Uygulamalar arası parametre gönderimi mecburi ise tüm sistemde mutlaka SSL sertifika yâda parametre kontrol teknikleri kullanılır.

5.8 Hata Yönetimi (Exception Management):

Oluşan tüm hatalar sadece genel bir hata mesajının dönmesi olarak algılanmasından ziyade, mutlaka hataların kayıt altında tutulması ve gerçek hataya sadece yöneticiler ulaşmasını sağlayacak sürecin oluşturulması gerekmektedir. (Hata mesajları trigger ile yöneticiye mesaj gönderen bir veritabanı tablosuna yazılabilir.)

5.9 Kayıt Tutma ve Denetim (Logging and Auditing):

Uygulama veya uygulamanın yöneticileri saldırı altında olduklarını anlayabilecekleri algoritmalar mutlaka oluşturulur. Bir uygulamaya ilişkin normal süreç ve şablon tanımlanır ve bunu dışında bir olay olduğunda saldırı ihtimali ele alınır.

Yukarıdaki ve bunlara benzer onlarca tehdit güvenilir uygulamalar geliştirmek için yazılım geliştirme sürecinin güvenliğinin yönetilmesinin büyük önem arz etmekte olduğunu gözler önüne sermektedir.

6. YAZILIM TESTİ

Test, bir sistemi manuel veya otomatik yollarla deneyerek veya değerlendirerek, belirlenmiş gereksinimleri karşıladığının doğrulanması veya beklenen ile gözlenen sonuçlar arasındaki farkların belirlenmesi sürecidir.

Yazılım testi ise bir yazılımın sonsuz sayıdaki çalışma alanından, sınırlı sayıda ve uygun şekilde seçilmiş testler ile beklenen davranışlarını karşılamaya yönelik, dinamik olarak yapılan doğrulama faaliyetlerini kapsamaktadır. Bu kapsamda dikkat edilmesi gereken hususlar şunlardır:

- **Dinamik olarak:** Yazılım mutlaka çalıştırılarak test edilmelidir.
- **Sınırlı sayıda:** Yazılımın neredeyse sonsuz sayıda olabilecek çalışma alanlarının tümünün testi imkânsız olacağından; kritiklik düzeylerine göre sıralanıp, yeterli görülen sayıda, en kritikleri test edilmelidir.
- **Uygun şekilde seçilmiş:** Test edilecek davranışın doğasına uygun ve muhtemel riskleri göz önünde bulunduran testlerin gerçekleştirilmesidir.
- **Beklenen davranışlar:** Test edilecek yazılımın, kullanıcı beklentilerine, gereksinimlerine ve akla uygun, mantıklı beklentilere cevap verebildiğinin test edilmesidir.

Gereksinimlere dayalı olarak uygunluk, tamlık, birlikte çalışma, hatalı girdi ve senaryo testlerini kapsayan testler işlevsel testlerdir. Performans ve güvenilirlik testlerini kapsayan testler ise işlevsel olmayan testlerdir. Karşılaşılacak kullanıcı davranışına, veri hacmine uygun, gerçekçi bir şekilde performans testleri şekillendirilmelidir. Güvenilirlik testleri ise olgunluk, hata-toleransı, toparlanma ile ilgili testleri kapsamalıdır.

Test faaliyetlerinin, yazılım geliştirme sürecinin daha başlangıç safhalarından itibaren vazgeçilmez bir parçası olduğu açıktır. Bu noktada yazılımların da bu faaliyetlere destek verir nitelikte olmasının önemi ortaya çıkmaktadır. “Test Edilebilirlik” kriterini önemli bir kalite kriteri olarak sunmaktadır. Bu kriterin sağlanması için, yazılım gerekleri tanımlanırken bu niteliği sağlayacak kalite gereklerinin belirlenmesi, geliştirme aşamasında da bu gereklerin sağlanması için çalışmaların yürütülmesi gerekmektedir.

Yazılımın tasarımı yapılırken, test planının da belirlenmiş olması, tasarımın test planına uygun özellikleri taşımasının sağlanması gerekmektedir. Bu sayede ileriki aşamalarda test edilebilirliğin sağlanması için ilk adım atılmış olacaktır.

A. GELİŞTİRME SÜRECİNDE YAZILIM TESTİ

Yazılım test faaliyetleri, tüm yazılım geliştirme süreci boyunca devam eden, sadece hataların bulunup ayıklanması işlemlerini değil, hata oluşmasını önleyici yaklaşımların uygulanmasını da içeren faaliyetlerdir. Gereksinimlerin belirlenmesi aşamasından itibaren başlayan bu faaliyetler yazılımların teslim aşamasına kadar farklı disiplinlere uygun olarak çalışılmasını gerektirmektedir.

Yazılım Geliştirme sürecinde test ile ilgili çalışmalara sistem tasarım tanımının ve yazılım gereksinim özelliklerinin hazır olmasını takiben yazılım test planı hazırlanarak başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

Yayınlanan sistem tasarım tanımı ve yazılım gereksinim özellikleri incelenerek, test konusundaki genel yaklaşım ve test faaliyetlerinin amacı göz önünde bulundurularak test aktivitelerinin planı, Yazılım Test Planı, hazırlanır.

Yazılım test planı çerçevesinde test platformlarının (test düzeneği), kontrolünü ve veri akışı bilgilerini, yazılım ve donanım test araçlarını (simülatörler, test sürücüleri) ve veri hazırlıklarını içeren test alt yapısı gereksinimleri belirlenir. Hazırlanan Yazılım Test Planı tanımlı sürece uygun olarak gözden geçirilir.

Testlerin planlanması kapsamında yazılım testlerinin hazırlık aşamasında, ortamın kurulmasında, ön testler sırasında, testlerin gerçekleştirilme aşamasında kimlerin sorumlu olacağı, testlerin gerçekleştirileceği ortamların planlanmasının nasıl yapılacağı, tanımlı süreç dışında projeye özel bir uygulama yapılıp yapılmayacağı, eğer farklılık varsa hangi aşamalarda süreçten farklılıklar olacağı, testler sonrasında raporlamanın nasıl ve kim tarafından gerçekleştirileceği, düzeltmelerin ve sonrasındaki testlerin nasıl yapılacağı gibi sorulara cevaplar verilir.

B. BİRİM TESTİ VE BİRİM ENTEGRASYON SÜRECİ

Yazılım test faaliyetleri arasında en alt seviyede birim testleri yer almaktadır. İleri aşamalarda yürütülecek test faaliyetlerinin öncesinde yazılan her kod parçasının doğruluğunun garantilenmiş olması gerekmektedir. Geleneksel yaklaşımda daha çok görsel denetimle ya da manuel yöntemlerle yapılan birim testleri, otomatik test araçlarının kullanımıyla daha sağlıklı ve tekrarlanabilir olarak yapılmaya başlanmıştır.

Test planının ve test tanımlarının hazırlanmasına paralel olarak, yazılımların kodlanmaya başlanması ile birim testleri gerçekleştirilmeye başlanmaktadır.

Yazılım Geliştirme Sürecinde Birim Testlerine yönelik olarak süreçte şu adımlar tanımlanmıştır:

- Yazılım Birim test hazırlıkları yapılır. Birim, modül ve bunların çeşitli bileşenlerinden oluşan kümelerin hangi sırayla test edileceği, test gereksinimleri (donanım ve yazılım), durumları ve yöntemleri belirlenir.
- Hazırlık kapsamında eğer gerekiyor ise sürücü/kart entegrasyonu ve bunların testi gerçekleştirilir. Kart seviyesinde koşan yazılımların ve geliştirilen veya hazır alınan kartın spesifikasyonlarına uygunluğu doğrulanır.
- Üretilen kod için birim/modül/küme testleri uygun debug/test araçlarıyla gerçekleştirilir. Testlerin kayıtları tutulur.
- Yeterlilik testi için yazılımın hazır olduğuna karar verilmesi durumunda kodun hazır olduğu duyurulur.

Birim testleri bu süreç dahilinde manuel ya da test araçları kullanılarak otomatik olarak yapılmaktadır. Test araçları kullanılarak otomatik test yapılması hedeflendiğinde, kullanılacak olan aracın kodlama aşamasına geçilmeden önce belirlenmiş olması ve aracın özelliklerine uygun şekilde yazılımın gerçekleştiriminin yapılması gerekmektedir. Otomatik test araçları yazılım birimleri için yazılan birim testlerinin işaretlenmesi ve bu testlerin dışarıdan erişilerek çalıştırılması mantığı ile çalışmaktadır. Bu araçlar test yordamlarının işaretlenmesini sağlayan işlevleri ve doğrulama cümlelerinin yazılabilmesini sağlayan (karşılaştırma işlemleri vs.) yordamları içeren kütüphaneyi ve testlerin gerçekleştirimini ve raporlanmasını sağlayan kullanıcı arayüzünü içermektedir.

Birim testi için manuel yöntemler kullanılması durumunda yazılım geliştirme ortamının hata ayıklama (debug) yeteneklerinden faydalanılmaktadır. Performansa yönelik testler için ise koda eklemeler yapılarak zaman bilgisi tutulmakta, yazılım işlevini tamamladıktan sonra bu bilgiler diske kaydedilerek değerlendirilmesi manuel olarak yapılmaktadır.

Yazılımlarda bellek sızıntısının olması durumu da sorun teşkil edebilecek önemli bir konudur. Birim testleri ile birlikte kod analiz araçları kullanılarak yazılımlar analiz edilmekte ve bellek sızıntıları tespit edilip düzeltilmektedir.

C. YAZILIM YETERLİLİK TESTİ

Yazılım yeterlilik testleri, kara kutu test yaklaşımıyla yazılımın, gereksinimlerini karşıladığının doğrulanması amacıyla yapılır.

i. Yazılım Yeterlilik Testine Hazırlık

- Yazılım gereksinim özellikleri temel alınarak ve testlerin gereksinimleri yeterli düzeyde kapsamasına özen gösterilerek test tanımları belirlenir.
- Testin başlatılması, yerine getirilmesi ve gerekli verilerin toplanması için gerekli olan adımların sıralamasını, her bir adım için beklenen sonuçları ve değerlendirme kriterlerini içeren test yönergeleri hazırlanır.
- Test tanımlarını ve yönergelerini içeren Yazılım Test Tanımları dokümanı tanımlı sürece uygun olarak gözden geçirilir. Bu gözden geçirme sürecinde kontrol listesi referans olarak kullanılabilir.

Yazılım yeterlilik test tanımlarının hazırlanması için öncelikle yazılımın gereksinimlerinin olgunlaşmış olması gerekmektedir. Gereksinimlerin tamamen veya kısmen belirsiz olması, test sürecinin sağlam temeller üzerine oturmasını engeller. Bir gereksinimin değişmez hale getirilip dondurulmasının pratikte zor olduğu düşünüldüğünde uygulama için önerilebilecek çalışma yöntemi, yazılımın gereksinimlerinin yazılımı geliştirenlerin deneyimleri doğrultusunda test edilebilecek olgunluğa erişmiş olduğu kanaatine varılması ile gereksinim dokümanının dondurulması ve bir kopyasının alınıp, gereksinimlerin test edilebilecek düzeye erişip erişmediğinin kontrolü için test tasarım yapan yazılım sorumlusunun görüşüne sunulmasıdır. Daha önceden aynı özelliklere sahip yazılımların testlerini muhtemelen gerçekleştirmiş olan yapan yazılım sorumlusunun her bir gereğin nasıl test edileceğini ve test sırasında nasıl bir ortamın sağlanması gerektiği konusunda bütün gereksinim dokümanını gözden geçirmelidir. Gereksinimlerin test edilebilir olgunluğa erişmiş olduğunu onaylandığında, dondurulmuş gereksinimlere göre test hazırlıklarına paralelden başlayabilir ve test hazırlıkları devam ederken bir yandan da gereksinimlerin olgunlaşması devam edebilir. Ancak test hazırlıklarının son aşamasında test tanımları ve gereksinimler arasında bir uyumsuzluk oluşup oluşmadığı kontrol edilmelidir.

Yazılımın gereksinimlerinin olgunlaşması ve test edilebilirliğe göre gözden geçirilmesi test tanımları hazırlanması aşamasına geçiş için olmazsa olmaz iki önkoşuldur. Test tanımları, doğruladığı gereksinimlerin yazılım tarafından karşılanıp karşılanmadığını bulmaya yönelik olmalı ayrıca olağan dışı girdileri de kapsayacak şekilde yazılmalıdır. Test tanımlarının yazılması sırasında dikkat edilmesi gereken bir diğer konu da her bir gereksinimin en az bir test tanımı ile doğrulanmasının gerekliliğidir. Gereksinim yönetim araçları kullanılarak gereksinimler ve test tanımları arasında izlenebilirlik kurulup test edilmemiş herhangi bir gereksinimin kalmaması sağlanmaktadır.

Hazırlanan Yazılım Test Tanımları dokümanında; testin amacı, testin hangi gerekleri adresleyerek doğruladığı, testin hangi konfigürasyonda gerçekleştirileceği, test yönergesi, testin başarılı sayılması için gerekli olan kriterler ve varsa test girdileri, önkoşulları ile çıktıları tanımlanmaktadır. Test yönergelerinin kapsamı ise doğrulanan gereğin kritikliği ve hata çıkma olasılığıyla doğru orantılı olmalıdır. Test sırasında çıkacak olası bir hatanın göreve etkisi ve sonuçları değerlendirilmelidir. Ayrıca testlerin belirlenmesinde; karmaşık, zaman içinde değişikliğe uğrayan, uygulama içinde yenilik olan (yöntem, yeni teknoloji vb.), optimizasyon gerektiren ve daha önce hata bulunan kısımlara özel önem verilmelidir.

ii. Test Yazılımları Geliştirme Faaliyetleri

Yazılım yeterlilik testleri için testlerin kara kutu yaklaşımı ile gerçekleştirilebilmesi, yazılımın sistemden bağımsız bir test ortamı içerisinde koşması için önem taşımaktadır. Yazılımda oluşabilecek hataları sistemde oluşabilecek hatalardan bağımsız olarak ayıklamak test işlemini kolaylaştırıp, verimliliği artırarak, hatanın olabildiğince erken aşamalarda yakalanmasını sağlar. Bu durum test edilecek yazılımın beraber çalıştığı her türlü yazılımın ve donanımın gerçek sistem üstünde çalışıyormuş gibi simüle edilmesi ihtiyacını doğurmaktadır. Test sistemi test yazılımlarından oluşmaktadır. Test yazılımları test edilecek yazılım ile simülasyonunu yaptığı yazılım veya donanımın ilgili arayüzünü kontrollü bir şekilde gerçekleyen yazılımlardır. Yani bir yazılımın testi sırasında o yazılımın haberleştiği yazılım veya donanımın simülasyonunun yapılması, o yazılım veya donanımın

tüm işlevlerinin gerçekleştirilmesi değil sadece test için gerekli olan arayüzün kodlanmasıdır. Öte yandan test yazılımları, ilgili arayüzleri gerçeklerken, gerçek sistemde çalışan yazılımlardan farklı olarak o arayüzdeki olağan dışı durumları da gerçekleme olanağına sahip olmalıdırlar. Bir yazılımın tam anlamıyla test edilmesi için zaman zaman birden fazla ve farklı yazılım geliştirme ortamlarında kodlanmış test yazılımlarına ihtiyaç duyulabilmektedir.

Gereksinimlerde yapılacak ufak bir değişiklik test yazılımlarında kökten değişikliklere yol açabilmektedir. Örneğin, test edilen yazılımın gereksinimlerinde olabilecek bir değişiklik test yazılımının tamamen başka bir ortamda tekrar geliştirilmesi gerekliliğini ortaya çıkarabilmektedir. Bu da işgücü ve zaman kaybına yol açabilmektedir. Bu yüzden testlerin planlama aşamasından sonra tasarımcı tarafından gözden geçirilmesi önem kazanmaktadır.

Test yazılımlarının hazırlanması aşamasının en önemli girdileri yazılım test tanım dokümanı ve yazılım arayüz tasarım tanımı dokümanlarıdır. Her iki dokümanın da olgunlaşması test yazılımlarının geliştirilmesinin başlaması için gereklidir.

Otomatik test yazılımlarının kullanılması ile testlerin daha hızlı gerçekleştirilmesi sağlanabilmektedir. Bu amaçla, kullanıcı arayüzünde yaptığı davranışları kaydederek daha sonra sanki kullanıcı yapıyormuş gibi aynı hareketleri farklı girdilerle defalarca tekrarlayabilen araçlar kullanılabilmektedir. Ancak farklı arayüzler üzerinden farklı ortamlara sahip çevre birimlerle etkileşimi çok olan gömülü yazılımların testi için bu gibi otomatik test araçlarının yetenekleri kısıtlı kalmaktadır. Bu gibi durumlarda pratik olarak önerilen yöntem, geliştirilmiş olan test yazılımı içerisine bu otomasyonun gömülmesi ve test yazılımı aracılığı ile testlerin mümkün olduğunca otomatik olarak gerçekleştirilmesinin sağlanmasıdır.

iii. Testlerin Gerçekleştirilmesi

Kodun hazır olduğunun duyurulmasını takiben yeterlilik testleri gerçekleştirilmeye başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

Test tanımları ve test araçları olgunlaştırılır.

- Yazılım gereksinimlerinden test tanımlarına izlenebilirlik kontrol edilir.
- Hedef prototip ve test yazılım ve araçlarının bir araya getirilmesiyle bir test düzeneği kurulur.
- Test tanımları çerçevesinde yeterlilik testleri gerçekleştirilir. Hata oluşması durumunda problemlerin çözülmesi amacıyla ilgili sürece geçilir ve süreç işletilir.

Testler gerçekleştirilmeye başlanmadan önce ortamın (test konfigürasyonunun) testlere hazır olup olmadığının kontrolünün yapılması gereklidir. Hazırlanan test yazılımlarının testler öncesinde doğrulanmış olması önem taşımaktadır. Resmi testler öncesi test edilecek yazılımın test ortamına entegrasyonun gerçekleştirildiği ve yazılımın testlere hazır olup olmadığının yazılımın sorumlusu tarafından kontrol edilir. Bu testler, test edilecek yazılımın en temel yeteneklerini doğrulayan test tanımları arasından seçilmelidir. Ancak bu testler başarılı olarak gerçekleştirilirse, resmi olarak testlere başlanmalıdır.

Testlerin hangi sırayla yapılacağı doğrulanan testlerin kritikliği ve testlerin fonksiyonel olarak birbirleri ile olan bağlantılarıyla alakalı olabilir. Bu sıranın ne olacağına test planlama aşamasında karar verilmeli ve onaylanmalıdır.

Yazılım sorumlusunun kendi testleri tamamladıktan sonra otomatik test yazılımı yapılır ve kurum içerisindeki görevli olarak tanımlanan kişi yazılımın yayınlanması, entegrasyon vb. işlemler için değerlendirir. Ayrıca ASÜ kendi bünyesinde olan yazılımlar ve satın alınan yazılımlar belirli aralıklarla düzenli olarak test edilir.

Yazılıma yapılan ekleme veya düzeltmeler yeni hatalara sebep olabilmektedir. Bu hataların tespit edilmesi ve

olası gerilemenin belirlenmesi amacıyla yapılan testler regresyon (gerileme) testleridir. Herhangi bir değişiklik sonucunda, yeni sürümü yapılan yazılımın regresyon testlerinde, sadece yeni sürümdeki değişiklik bilgisinin incelemesi sonucunda gerekli olduğunu belirlenen testler yapılabilir.

iv. Test Sonuçlarının Raporlanması

Test sonuçları Yazılım Test Raporları ile raporlanmakta ve ilgili kişilere duyurulmaktadır. Bir yazılım hatası, yazılım sorumlusu tarafından incelenmekte, yazılımda gerekli düzeltme yapıldıktan sonra girilen hatanın düzeltildiği bildirilmektedir. Yazılım sorumlusu üzerinde ilgili testi tekrar gerçekleştirerek hatanın giderilip giderilmediğini doğrulamaktadır.

Yeterlilik testlerinin tamamlanmasının ardından test kayıtları kullanılarak Yazılım Test Raporu hazırlanır.

D. SİSTEM ENTEGRASYON TESTİ

Yazılım yeterlilik testlerinin gerçekleştirilmesi sonrasında sistem tarafında da çalışmalar entegrasyon testi hazırlıkları aşaması ile devam eder. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Her bir entegrasyon aşamasındaki gereksinimlere uygun olarak test tanımları yazılımın sorumlusu tarafından belirlenir ve gerekli test araçları oluşturulur. Tasarlanması gereken özel amaçlı donanım / yazılım test araçları ve yeni test yatırımları için donanım geliştirme sürecine uygun olarak planlama yapılır.
- Test tanımlarının dokümante edildiği dokümanlar tanımlı sürece uygun olarak gözden geçirilir.
- Tester olarak tanımlanan personeller tarafından bütünleşik sistem elde edilene kadar ihtiyaca göre aşamalı olarak entegre ve test edilir. Entegrasyon testleri kapsamında öncelikle donanım ve yazılım arayüzlerinin doğruluğu kontrol edilir. Arayüzler doğrulandıktan sonra çalışma senaryoları kapsamında fonksiyonel ve performans testleri yapılır.

Otomatik test yazılımı yapan program tarafından test sonuçları raporlanır. Bu Test sonuçlarına göre yazılım sorumlusu tarafında düzeltmeler yapılır.

E. SİSTEM YETERLİLİK TESTİ

Sistem entegrasyon ve testleri tamamlandığında Sistem İşletme Testlerinin yapılması ile sistemin doğrulanması sağlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Sistem İşletme Test Planı içinde yer alan test gereksinimlerinin sağlanmasında kullanılacak olan test tanımları yazılır.
- Sistem, gerçek kullanım ortamında veya benzer koşullarda hazırlanan test tanımlarına uygun olarak test edilir ve rapor oluşturulur.
- Raporlar Gözden Geçirme Sürecine göre gözden geçirilir ve doğrulanmış ve geçerli kılınmış sistem elde edilir.

Sistem yeterlilik testlerine yönelik çalışmalar Sistem Gereksinim Özellikleri (SGÖ) dokümanı hazırlandıktan ve gözden geçirme süreci tamamlandıktan sonra başlar.

Sistem İşletme Test Tanımı (SİTET) dokümanına her bir gereksinim için doğrulama metoduna uygun olarak test tanımları yazılır ve testler uygulanır.

Yazılım tasarımı yoğun olan sistemlerde sistemde doğrulanması entegrasyon testleri sonrasında tamamlanmasına rağmen yazılım ile ilgili gereksinimlerin doğrulanması sistem işletme testleri sonunda tamamlanmaktadır.

F. SİSTEM KABUL TESTİ VE SİSTEM SAHA TESTİ

Tasarım ve doğrulama süreci tamamlanan sistemin kullanıcının isteklerini karşılayıp karşılamadığı kabul testleri ile ispat edilir. Müşteri tarafından belirlenmiş bütün gereklerin doğrulanması için Muayene Kabul Dokümanı oluşturulur.

Sistem bu dokümana göre önce kurum içerisinde doğrulandıktan sonra kullanacak personel davet edilir ve testler müşteri ile birlikte tekrarlanır.

Sistem kabul testlerinin, sistem özelliklerine bağlı olarak kurum içerisinde ve/veya arazide yapılması gerekebilir.

DÜZELTİCİ ÖNLEYİCİ FAALİYETLER

Doğrulama faaliyetlerinin önemli bir parçası hatayı oluşmadan önlemeye yönelik çalışmaların yürütülmesidir. Bu çalışmalar, ileride ortaya çıkabilecek hataları önleyerek, test ve düzeltme maliyetlerini azaltmayı hedeflemektedir. Bu kapsamda yapılan başlıca faaliyetler gözden geçirme ve tasarım doğrulaması faaliyetleridir.

Gözden geçirme, sürecin her safhasında farklı ürünler için uygulanabilen bir yöntemdir. Kurumumuzda yazılım geliştirme sürecinde hazırlanan dokümanlar, yazılım mimari tasarımları, detaylı tasarımlar ve yazılan kodlar için gözden geçirme gerçekleştirilmektedir.

Gözden geçirme çalışmaları için süreçte tanımlanan temel adımlar şunlardır:

- Ürünün gözden geçirmeye çıkarılması: Gözden geçirmeye hazır olduğu düşünülen ürün üzerinde değişiklikler dondurularak, gözden geçirme için belirlenen ekibe duyurulur. Belirlenen süre içerisinde ürünün incelenmesi ve gözden geçirme kaydının doldurulması istenir. Bu kayıtlar doldurulurken ürün için hazırlanmış olan kontrol listeleri de referans alınır.
- Gözden geçirme kayıtlarının incelenmesi: Belirtilen süre sonunda tüm gözden geçirme kayıtları ürünü hazırlayan kişi tarafından toparlanır ve kayıtların görüşülmesi için toplantı düzenlenir. Toplantı sırasında kayıtlar üzerinden geçilerek ürüne yansıtılacak değişiklikler belirlenir.
- Gözden geçirme kayıtlarının işlenmesi: Gözden geçirme toplantısı sırasında alınan kararlar doğrultusunda üründe değişiklikler yapılır.

Bu çalışmalar, ürünlere farklı paydaşların gözüyle bakarak erken safhalarda hataya sebep olabilecek hususların belirlenmesini ve düzeltilmesini sağlar. Hatanın önlenmesine yönelik diğer faaliyetler tasarım doğrulama faaliyetleridir. Bu faaliyetler kapsamında, tasarım yapıldıktan sonra sistem senaryoları ele alınarak tasarımın tüm senaryoları karşılayıp karşılamadığı değerlendirilmektedir. Kullanıcı arayüzü tasarımlarının doğrulanması için ise hızlı prototipleme çalışması yapılarak ekranlar oluşturulmaktadır. Bu ekranlar üzerinde, yine senaryolar çalıştırılarak, işlevlerin akış sırasının ve ekranların amacına uygunluğu gözden geçirilir.

G. SONUÇLAR

Sistem ve Yazılım geliştirme sürecinde doğrulama çalışmaları kurumumuzda tanımlı süreçlere uygun olarak yürütülmektedir. Yazılım test faaliyetlerinin disiplinli bir biçimde yürütülmesi, sistemin hatasız çalışmasını sağlamaktadır. Bu yüzden testlerin planlama aşamasından, hazırlanma, gerçekleştirme ve raporlanma aşamalarına kadar geçen sürede uzman kişiler rol almakta, basit ve anlaşılır iş talimatları ile test süreci desteklenmektedir. Testler yapılırken mümkün olduğunca hazır veya projeye özgü geliştirilen test yazılımlarından destek alınmaktadır. Hatalar, hata takip araçlarından takip edilmektedir.

7. SORUMLULUK

Bu prosedürün uygulanmasından kapsam dâhilindeki tüm personel sorumludur.