YAZILIM TEST MÜHENDİSLİĞİ

Dr. Öğr. Üyesi Hasan YETİŞ

Yazılım Test Yaşam Döngüsü

- Geliştirilen yazılımlar ve **güncellenen sürümlerde** ortaya çıkan her hata, yazılımın kalitesini ve işlevselliğini olumsuz etkiler.
- Yazılım yaşam döngüsü boyunca geliştirilen **sistemin kalitesini arttırmak** için test yaşam döngüsü büyük önem taşımaktadır.
- Test yaşam döngüsü ile
  - sistemde var olan **hatalar tespit edilir**.
  - Gerçekleştirilen test işlemi daha sonra bu hataların bulunduğu modüller, **nedenleri ve çözümleme yöntemleri** raporlanarak sistemin ileri aşamalarında hataların tekrar etmesi durumunda kullanılır.
- Test işlemi hataların çözümlenmesinin yanı sıra **sistem çalışabilirliğini kontrol etmek, gereksinimleri değerlendirmek, güncellemeleri ve son sürümün hata oranını kontrol etmek** için de gerçekleştirilir

Yazılım Test Yaşam Döngüsü Ve Süreçler

- Yazılım testi; “Ürün kullanıcılarının ihtiyaçlarının karşılanma oranını belirlemek için yapılan bir hata tespit işlemi” olduğu söylenebilir.
  - Uygun test tekniklerine göre **test seviyeleri** ve **test türleri** belirlenmelidir.
  - Hatanın bulunduğu kod blokları **tespit edilmeli** ve işaretlenmelidir.
  - Tespit edilen hata, önem derecesi ve aciliyetine göre **sınıflandırılmalıdır**.
  - Hatanın çözümlenebilmesi için **öncelik durumu** dikkate alınmalı, kullanılacak yöntemler doğru bir şekilde belirlenmelidir.
  - Test süreçlerinde en uygun **yazılım araçları belirlenmeli** ve gerekli aşamalarda etkili kullanılabilmelidir.

DOĞRULAMA	GEÇERLEME
➢Ürünü doğru mu üretiyoruz? ➢Sistemin hatasız ve iyi bir mühendislik ürünü olup olmadığını ölçer. ➢Geliştiriciler veya QA ekibi tarafından gerçekleştirilir. ➢Doğrulama aşamasında bulunan hataların maliyeti daha azdır.	➢Doğru ürünü mü üretiyoruz? ➢Sistemin kullanıcı gereksinimlerine uygunluğu ölçer. ➢Test ekibi tarafından gerçekleştirilir. ➢Geçerleme aşamasında bulunan hataların maliyeti daha fazladır.

Yazılım Test Yaşam Döngüsü Ve Süreçler

- Test işlemleri, yazılım geliştirme yaşam döngüsünün her aşamasında aslında gerçekleştirilmektedir.
- Her aşamada birbirinden farklı çok sayıda test işlemleri ve süreçleri yer almaktadır.
  - *Planlama* sürecinde teknik personellerin sayısı ve teknik yeterliliklerinin **doğrulanması**
  - *Çözümleme* sürecinde geliştirilen UML diyagramlarının **doğrulanması**
  - *Tasarım* sürecinde gerçekleştirilen fiziksel arayüzlerin **doğrulanması**
  - Kodlama sürecinde proje kodlarının test edilmesi ve **doğrulanması**
- Ayrıca, yazılım geliştirme süreci tamamlandıktan sonra da test süreci devam edebilir

Test ile;

- Yazılım test yaşam döngüsü süreci ile sistemin **hata oranı belirlenir** ve buna bağlı olarak yazılımın kaliteli olmasına engel olan her durum ortadan kaldırılır.
- Test sonuçlarına göre **sistem ile ilgili bilgi, görüş ve önerilerde** bulunulur.
- Yazılımın kriterlere uygunluğu, **performansı, doğruluğu ve geçerliliği** ölçülür.
- Sistemde bulunan ve **hata riski barındıran modüller**, test edilip raporlanır. Sonraki aşamalarda hata riski barındıran modüller tekrar test edilerek olası hataları ortadan kaldırılır. Bu nedenle testler, yazılımın kalitesini arttırmada önemli rol oynar.

Test ile;

Sistemle ilgili her türlü **eksiklik veya kusurlar** belirlenir;

- Sistemin **planlanan** zaman, maliyet ve kriterler **doğrultusunda ilerlemesi** kontrol edilir.
- Sistemde var olan **yollar kontrol** edilerek sistemin doğruluğu ortaya konulur.
- Sistemin beklenilen hız, alan gibi özellikleri kontrol edilerek **daha performanslı** çalışması hedeflenir.
- Sistemde var olan **güvenlik zaafiyetleri** tespit edilerek, sistem açıkları giderilir.
- Sistemin **kullanıcı dostu** olması açısından kullanılabilirlik testleri uygulanır.
- Sistemin olumsuz durumlardaki tavrı kontrol edilerek, bu tür durumlar için gerekli çözümler üretilir.
- Sistemin **farklı cihazlarda** veya işletim sistemlerinde kusursuz çalışması için gereken testler yapılır.

Test Türleri

- Yazılımın kalitesi açısından önemli olan test işlemleri, **manuel** ve **otomatik** olarak iki şekilde gerçekleştirilebilir.
- Manuel test süreci, hataların bulunması için test uzmanlarının **test senaryolarını** çalıştırıp denetlemesi sürecidir.
- Bu süreçte, test uzmanları **son kullanıcı rolü** ile sistemi test eder ve herhangi bir otomasyon aracı olmaksızın **raporları manuel** olarak oluşturur.
- Bir diğer test gerçekleştirme işlemi **otomatik** testtir. Otomasyon testi ile hataların bulunması için bir yazılım otomasyon aracından yardım alınır. Bu süreçte otomasyon araçları ile otomatik olarak çalıştırılan **test komut dosyaları** sayesinde test sonuçlarına ulaşılır.

Test Türleri

- Yazılım testi, işlevsel (fonksiyonel) ve işlevsel olmayan (fonksiyonel olmayan) test olmak üzere iki ana başlık altında incelenebilir.
- ***İşlevsel test türleri***, yazılım fonksiyonlarının gereksinimlere uygunluğunun doğrulanması için gerçekleştirilen testleri içermektedir. Test sonunda elde edilen çıktının beklenen çıktıyla eşdeğer olup olmadığı kontrol edilir. Fonksiyonların **girdi-çıktı değerlerinin** kontrol edildiği işlevsel testlerde uygulamanın kaynak kodu denetlenmediği için Kara Kutu testi kapsamında yer almaktadır.
- ***İşlevsel olmayan test türleri*** ise sistemin işleyişinin performans, yük, ölçeklenebilirlik, güvenlik, uyumluluk vb. açısından denetlendiği test türüdür. Esas amaç, sistemi bir talep karşısında ne kadar hızlı yanıt verdiğinin değerlendirilmesidir. Ayrıca test esnasında sisteme yöneltilen istek doğrultusunda güvenlik, yük, stres gibi özellikler de kontrol edilir.

Yazılım Test Yaşam Döngüsü

Şekil 3.2. Yazılım test yaşam döngüsü

İhtiyaç Analizi

- Test yaşam döngüsünün ilk aşaması olan ihtiyaç analizi, **test işlemlerinin sağlıklı** ve **verimli** bir şekilde gerçekleştirilmesi ve hedeflenen test çıktısına ulaşılması için büyük önem taşımaktadır.
- Test yaşam döngüsünde ihtiyaç analizi, **test edilmesi gereken** modüllerin ve özelliklerinin tanımlandığı aşamadır.
- Test işleminde hedeflenen sonuca varmak için **test öncelikleri ve odak noktalar** belirlenir.
- Test yaşam döngüsünün ilk aşamasında, Kalite Güvence ekibi tarafından, test edilecek sistem ve test için **dikkate alınması gereken gereksinimler** listelenir.
- Bu aşama, **kontrol edilmesi gereken modüllerin, yapılacak testlerin türleri** ve **seviyelerinin** analizi için gereklidir.

Test Planlama

- Test planlama aşaması, test yaşam döngüsüne dair tüm stratejilerin belirlendiği kısımdır.
- Testin **amacı, kapsamı, test stratejisi ve seviyesi, riskler, test sonucunda ulaşılabilecek hedefler ve testin maliyeti** bu aşamada belirlenir. Bu aşamada, test ekibi tarafından belirlenmesi gereken önemli hususlardan bazıları aşağıdaki gibidir:
  - Yazılım sistemini analiz etmek,
  - Testlerin amaç ve kapsamını belirlemek,
  - Yapılacak test türlerinin belirlenerek, test stratejileri oluşturmak,
  - Testlerin özelliklerine göre önemli kriterleri belirlemek,
  - Testler için önemli ve uygun kaynakları belirlemek,
  - Tüm projeyi test etmenin maliyetini hesaplamaktır.

Test Senaryolarının Oluşturulması

- Test planlamasın sonra test senaryolarının çalışması başlar.
- Test senaryosu, sistemin gereksinimleri karşılayıp karşılayamadığı ve doğru çalışıp çalışmadığını belirlemek için test sırasında kullanılan **bir dizi işlem sıralaması** olarak tanımlanabilir.
- Bu aşamada test ekibi tarafından, testlerin yürütülmesi için test senaryoları ayrıntılı olarak tasarlanır.
- Test senaryolarının oluşturulmasında sistemi içeren her türlü husus göz önüne alınarak uygun ve birbirinden farklı yaklaşımlar hazırlanmalıdır.
- İyi bir test senaryosu, basit ve açık işlem adımlarını içerecek şekilde yazılmalıdır.

Test Ortamının Hazırlanması

- Test ortamı, **test senaryolarının yürütülmesi** için hazırlanan yazılım ve donanımları içermektedir.
- Bu aşamada test ekibi, bütün test senaryolarını yürütebilmek için yazılım, donanım gibi **sistemsel kaynakları** kurmakla görevlidir.
- Test ortamında, **ürünün test edileceği koşullara karar verilir**. Birden çok test ortamının olması mümkündür.
- Test sunucusunun kurulumu, ağ kurulumu, bilgisayar, tablet ya da mobil cihaz kurulumu bu aşamada gerçekleştirilir. Test ortamının düzenli ve eksiksiz olması test işleminin kaliteli şekilde gerçekleşmesi ve doğru sonuçlar üretilebilmesi için önemli bir adımdır.

Testin Yürütülmesi

- Bu aşamada test işlemi kod ile yürütülür. Test işlemi sonucu **beklenen değer ile çıkan değer karşılaştırılır** ve raporlanır
- Raporlanan değerlerin analizi de bu adımda gerçekleşir.
- Test senaryolarına ve iş-akış planlarına bağlı kalarak, manuel veya otomatik olarak sistemin **muhtemel hatalarının fark edildiği aşamadır**.
- Her bir test durumu sonlandığında, **'Başarılı', 'Başarısız'** veya **'Engellendi**' olarak işaretlenir. Test **senaryolarında bulunan hatalar** sebebiyle engellenen belirli senaryolar 'Engellendi' olarak işaretlenir.
- Testlerin başarısız sonuçlanması durumunda ilgili kusurlar hata izleme sistemleri yardımıyla yazılım **geliştirici ekibine bildirilir**.

Testin Raporlanması

- Son aşamada, **test lideri tarafından bir test döngüsü kapanış raporu** hazırlanır.
- Raporlama aşamasında, mevcut test sürecinde karşılaşılan sıkıntılı durumlar belirtilerek, **yeni test döngülerinin iyileştirilmesi** adına referans olarak kullanılabilir.
- Süreç, proje boyutuna ve hataların riskine göre değişiklik gösterir.
- Test lideri tarafından hazırlanması gereken raporda
  - test sürecinin özeti,
  - test boyunca kullanılan tanımlayıcılar,
  - testin kapsamlılık değerlendirmesi,
  - sonuçların özeti ve değerlendirilmesi,
  - faaliyetlerin özeti ve
  - kalite güvence ekibi tarafından verilen onay bulunmalıdır.
- Test kapanış aktivitelerinde en önemli iki unsur**, hata analizi** ve **test özetinin** sağlıklı bir biçimde hazırlanmasıdır. *Hata analizi*, var olan hataların **öncelik ve risk** ortamlarını, **bulunduğu fonksiyonları** ve **çözüm yöntemini** içerir. Test *özeti* ise **testin amacı**, **test stratejileri ve metotları**, test esnasında üzerinde **yoğunlaşılan modüller**, test **sonuçları** ve testlerin **analizidir**.

Testlerdeki Süreklilik Durumları

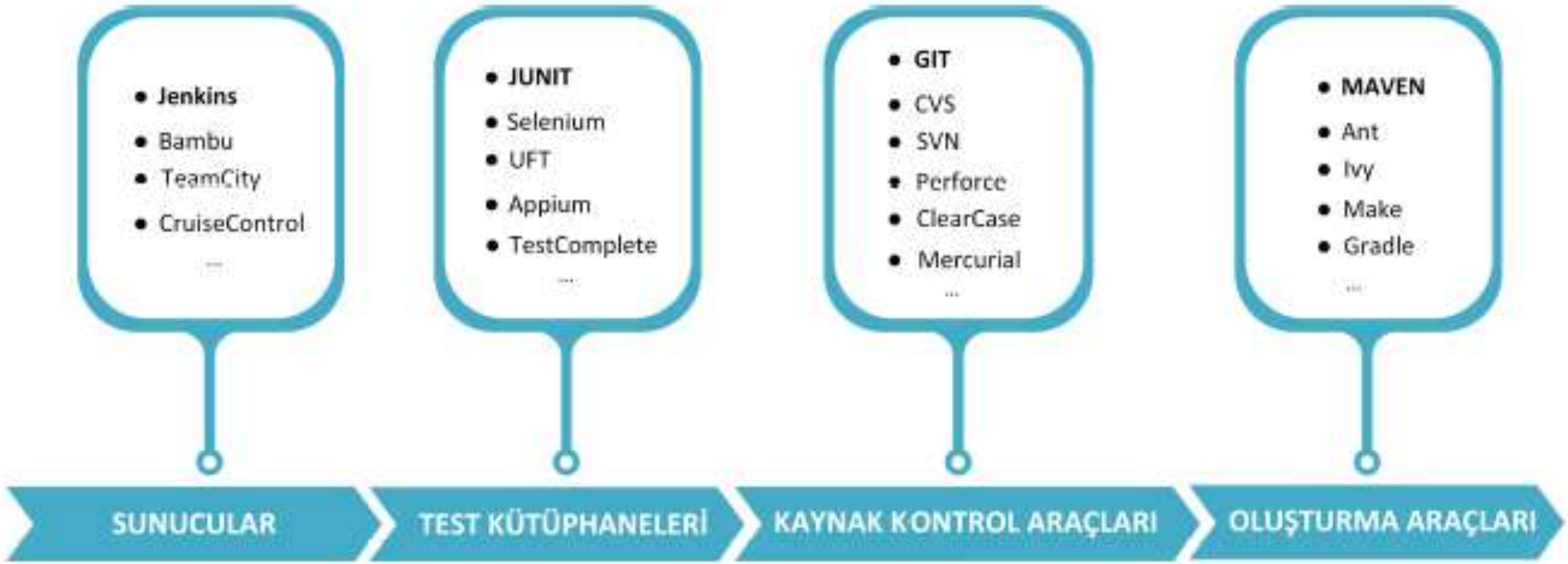
- Yazılımda **hataların erken tespit edilmesi, hızlı çözülebilmesi** ve sistem işleyişini etkilememesi için farklı yaklaşımlar uygulanmaktadır. Bu yaklaşımlar
  - **sürekli entegrasyon,**
  - **sürekli teslimat ve**
  - **sürekli dağıtımdır**.
- Bu yaklaşımlar sistem sürümlerindeki değişikliklerin yayınlanmasında ortaya **çıkabilecek hataların sayısını** ve etkisini azaltmaktır.
- Sürekli teslimat, sürekli entegrasyon ve sürekli dağıtım işlemleri, **Çevik yazılım geliştirme yöntemlerinin ana manifestosu** içerisinde yer alan en temel ve önemli özelliklerdendir.

Sürekli Entegrasyon

- Günümüzde, şirketler bu süreçleri aktif olarak kullanmaktadırlar. Bu süreçlerin önemsenmesi, müşteri memnuniyetini arttırarak **iyileştirilmiş yazılımın geçerliliğini** sağlamaktadır.
- Sürekli entegrasyon işlemleri, **kod üzerindeki değişikliklerin sürekli bir şekilde izlenmesi, derlenmesi ve test edilmesi** için tasarlanan işlem adımları bütünüdür.
- sürekli entegrasyon, **kod bloklarındaki her değişikliği otomatik olarak test etme mantığına dayanır.** Bu mantıkla, kaynak kodların devamlı gelişerek yeni sürümler olarak yayınlanması sağlanır.
- Sürekli entegrasyon yaklaşımı ile yazılımdaki herhangi bir hatanın varlığını tespit etmek için **otomasyon testi esas alınır**. Genellikle **birim** testleri kullanılarak gerçekleştirilir.

Sürekli Entegrasyon

- Geliştiriciler her an **kodlarını derleyerek sürüm oluşturma** isteği gönderebilirler. Bu yöntem ile geliştirilen her bir entegrasyonun, otomatik bir yapı sayesinde hatalarından arındırılması amaçlanır.
- Sürekli entegrasyonda geliştiriciler tarafından doğrulanan sistem, **derlenir ve hemen ardından test edilir**.
- Yapılan test işlemi sonucu hatanın tespit edilmesi otomatik bir şekilde gerçekleştirilir.
- Test süreci başarılı olursa yazılan kod mevcut **sisteme entegre edilir**. Doğrulama, derleme, test işlemleri birbirini takip eden tekrarlı bir süreç olduğundan bu yöntem sürekli entegrasyon olarak isimlendirilir.



Şekil 3.4. Sürekli entegrasyonda kullanılan araçlar

Sürekli Entegrasyon Adımları

1. Geliştiriciler tarafından yazılan **kodlar sunucuya** aktarılarak değişiklikler yapılır. Sunucudaki kod değişiklikleri izlenir ve kontrol edilir.
2. Sunucudaki kodlar **derlenerek sistem çalıştırılabilir hale getirilir**.
3. Daha sonra sırayla birim ve entegrasyon **testleri yürütülür**.
4. Testleri başarıyla sonuçlanan kodlar **doğrulanarak kabul edilir**.
5. Sürüm(versiyon) sistemi yardımıyla derlenen kodlara **sürüm numarası ve etiket atanır**.
6. Ekip sistemin son durumu hakkında bilgilendirilir. Projede yer alan personellere sistemin son durumu hakkında bilgi verilir. **Başarısız testler** olursa geliştirici ekip yeniden uyarılır. Böylece ekibin hatalı kod bloklarına hızlı bir şekilde müdahale etmesi sağlanır.

Sürekli Entegrasyon

Sürekli Entegrasyon Olmadan Geliştirme	**Sürekli Entegrasyon İle Geliştirme**
Hata sayısı daha fazladır.	Hata sayısı azdır.
Entegrasyon sıklığı azdır.	Entegrasyon sıklığı fazladır.
Sürümler az sıklıkta ve geç yayınlanır.	Sürümler düzenli bir şekilde ve hızlı yayınlanır.
Test süreci, zorludur ve uzun sürer.	Test süreci, kolaydır ve hızlı sonuçlanır.
Raporlama süreci zor ve yavaş geçer.	Raporlama süreci hızlı, kolay ve verimli geçer.
Çıktı için yazılımın sonuçlanması beklenir.	Daha şeffaftır ve anında proje çıktısı yayınlanır.

Sürekli Entegrasyon

- Sürekli entegrasyon sürecinin aşağıda belirtildiği gibi birçok avantajları bulunmaktadır.
  - Geliştirilen **yazılımların kalitesinin** arttırılarak nitelikli olması sağlanır.
  - Geliştiricilerin **bağımsız ve paralel olarak çalışmasına** olanak sağlar.
  - Gerçekleştirilen testlerin **tekrarlanabilir** olmasına yardımcı olur.
  - Derleme işleminin **tam otomatik** bir biçimde gerçekleştirilmesi imkanını sunar.
  - Dağıtım işlemleri **daha hızlı** gerçekleştirilir.
  - Hata olduğunda geliştiricilere giden bildirim ile **entegrasyon** sırasında ortaya çıkabilecek **hataların azaltılması** sağlanır.

Sürekli Entegrasyon

- Bunların yanı sıra bir dizi de dezavantajları bulunmaktadır.
  - Sürekli entegrasyon araçlarının **kurulumu ve bu konuda personelin eğitimi** için süre gereklidir.
  - Yazılan testlerin **prosedürlere uygun** bir şekilde oluşturulması gerekmektedir.
  - **Kapsamlı testlerin gerçekleştirilmesi** sürekli entegrasyon sunucusu için fazla sayıda **kaynak** gerektiren bir işlemdir.
  - Birden fazla geliştirici tarafından yazılan kodların sisteme eş zamanda entegre edilmesi durumunda **bekleme süreleri** uzayabilir.

- **Sürekli Teslimat:** Geliştirme sürecindeki yazılım, sık sık küçük değişimler ile güncellenir. Buna bağlı olarak ürünün teslim süresi kısalır. **Sürekli teslimat sürecinde kod yayınlanmaya her an hazırdır.** Sürekli teslimatta, kodun yayınlanması tetiklenmeli veya belirli periyotlarda ayarlanmalıdır
- **Sürekli Dağıtım:** Sürekli dağıtımda **testi başarıyla geçen her kod doğrudan, otomatik bir şekilde yayına alınır**. Tamamen otomatikleştirilmiş bir sistemle gerçekleşen bu süreci, yalnızca kod testinin başarısız sonuçlanması engeller. Sürekli dağıtım ile yapılan her değişiklik doğrudan kullanıcının kullanımına hazır hale gelir.

Özellik	Sürekli Entegrasyon	Sürekli Teslimat	Sürekli Dağıtım
Kod bloklarındaki değişikliğe göre otomatik test	X	X	X
Sürüm halinde devamlı gelişen kaynak kod	X	X	X
Erken aşamalarda gerçekleşen hata tespiti ve daha kolay çözümü	X	X	X
Ortaya çıkan sürümlerin yayınlanmaya hazır olması		X	X
Sürüm halindeki kaynak kodun otomatik yayına alınması			X
Yapılan her değişikliğin doğrudan kullanıcının kullanımına açılması			X
Yeni özellik ve eklentilerin daha hızlı dağıtılması ve doğrulanması			X