Yazılım Geliştirmede Çevik Yöntemler

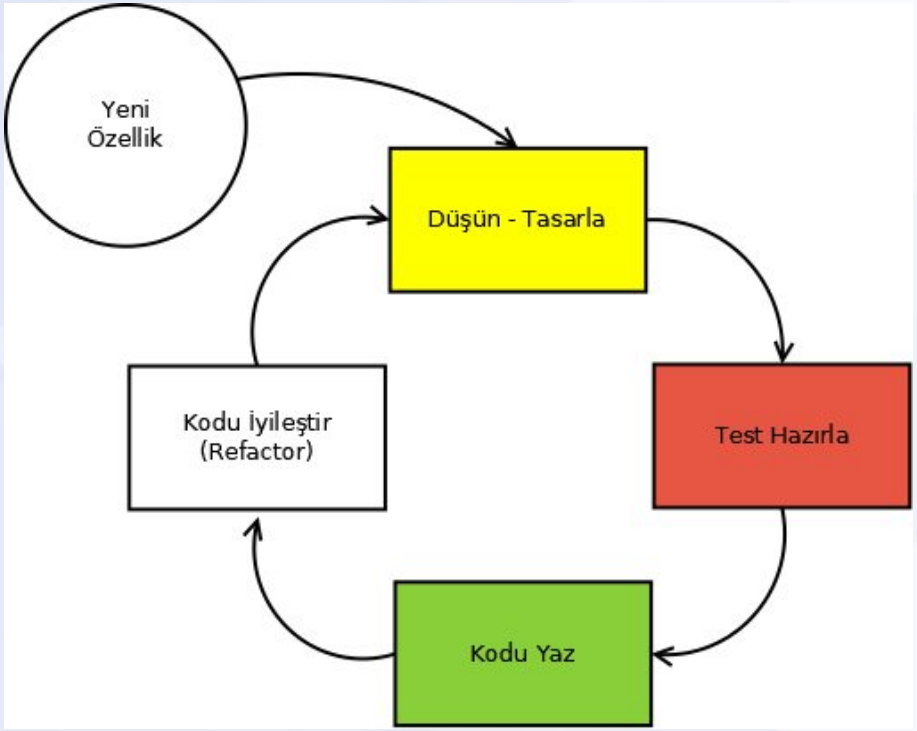
Hafta 5

Test Güdümlü Geliştirme

Sibel Kuzgun Akın

Test Güdümlü Geliştirme

- *Test Driven Development*
- *Süreç yeni özellik isteğiyle* başlar.
- Önce test verileri (girdiler – beklenen çıktıları) hazırlanır.
- Sonra kod yazılır.
- Temiz koda ulaşmak için kod iyileştirilir.
- Kodlama bir döngü içinde sürer.



Kısa Kısa...

- 2003'te Kent Beck tarafından ortaya atıldı.
- Uç programlamanın unsurlarından biridir.
- Daha sonra kendi başına bir yaklaşım olmuştur.
- Önce ihtiyaç analizi yapılır.
- Sonra buradaki kullanıcı öykülerine dayanarak, test verileri tanımlanır.

Nasıl çalışır?

1. Programa yeni bir özellik eklemeden önce yapılacak testlerin listesini hazırla.

2. Listedeki bir testi önceden kalan test durumlarına ekleyip, mevcut programda otomatik test ile dene.

3. Yeni eklenen özelliğin testleri başarısız olmalı! Çünkü daha özellik eklenmedi. :)

4. Yeni testleri geçebilecek en basit kodu yaz.

5. Şimdi bütün testler geçilmeli.

6. Gerekiyorsa kodu iyileştir ama işlevselliğini koru. Yeniden test et.

7. Listede sıradaki test ile 2. adıma dön.

Bu döngü programa eklenen her yeni özellik için tekrarlanır.

Kullanılan kütüphanelerde hatalar olduğundan şüphe edilmiyorsa, onları test etmeye gerek yoktur.

Test Verileri

- Test verileri: Olası girdiler ve beklenen çıktıları
- Beyaz kutu testleri yapar: Program koduna erişimin olduğu durumlarda kullanılabilir.
- Nesne yönelimli programlamada (OOP) bilgi saklama (information hiding) ilkesi nedeniyle sınıfları yazanlar testleri yapmalıdır.
  - Sınıfa özel (private) veri ve işlevlere erişim için Java, .NET, C++ gibi programlama dilleri bazı kolaylıklar sunmaktadır.
  - Sınıfa özel işlevlerin TDD ile test edilmesine gerek var mı, yoksa sınıfın herkese açık arayüzünü test etmek yeterli mi, tartışılmaktadır.

Test Türleri

- Birim test *(unit test)*: Bir işlevi, modülü, program parçasını test eder.
- Eğer birim test ile denenmek istenen kod başka kütüphaneleri, vb. kullanıyorsa bu aslında entegrasyon testine dönüşür.
- Dışarıdaki bir veri tabanı, web servisi, vb. kullanılıyorsa, kodun birim testlere uygun hale getirilecek şekilde bölünmesi önerilir.
  - Bağımlılığı Tersine Çevirme İlkesi (Dependency Inversion Principle)
- Sahte (fake) veya taklit (mock) veriler, önyüz (frontend) programlarını denerken yaygın olarak kullanılır.
  - Örnek: Mock Turtle ile Javascript programları için taklit veriler hazırlayıp, önyüzü test etmek. https://mockturtle.net/

Kod Geliştirme Stili

- KISS: Keep It Simple Stupid
- YAGNI: You Aren't Gonna Need It
- Fake it till you make it. (Kent)
- Yazılımcılar programı yazmaya başlamadan önce onu nasıl test edeceklerini düşünmelidir.

Test Durumu

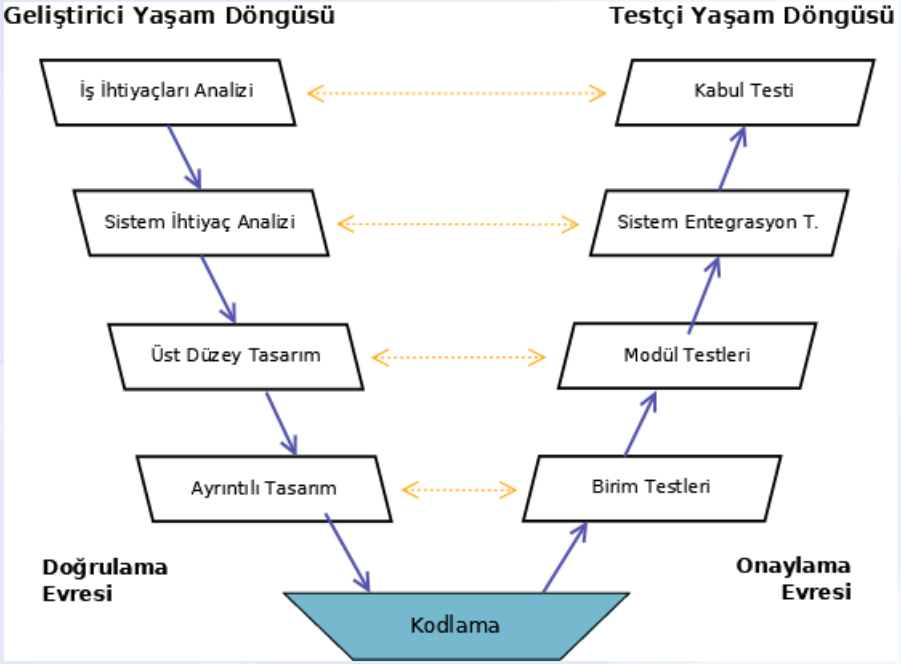
- Test durumu: Adım adım izlenecek bir talimat veya belirli girdilerden oluşur.
- Girdilerle beraber, test sonunda beklenen çıktılar tanımlanır.
- Örnek: Kullanıcı 16 rakamdan oluşan kredi kartı numarasını, son kullanma tarihini ve güvenlik kodunu girip onay düğmesine basarsa, ekran banka onay sayfasına yönlenecek.
- Bir işlevi veya uygulamanın bir özelliğini test etmek için, test durumlarını kullanmak uygundur.
- Test senaryolarından daha ayrıntılı olurlar. Girdi değerlerini ve beklenen çıktıları içerirler.
- Bir uygulamanın işlevlerinin doğrulanması için uygundur.

Test Senaryosu

- Bir uygulamanın nasıl çalışacağının üst düzeyden tarifidir.
- Bir programın veya programdaki bir özelliğin beklenen sonuçlarını test etmek için, adım adım yapılacakları tanımlar.
- Bir e-ticaret sitesi için “kullanıcı çiçek sepeti aratıp satın alabilir” bir senaryodur. Daha çok kullanıcı öykülerine benzer.
- Bir senaryo, birçok test durumu içerebilir.
- Örneğin, çiçek sepeti için çeşitli arama terimleri veya farklı çiçekler içeren sepetler bir senaryoda kullanılabilecek farklı test durumları olabilir.
- Uygulamadaki bir fonksiyon veya özelliğin nasıl çalışacağı ile ilgili bir fikir edinmek için test senaryoları kullanılır.
- Senaryolar, test durumlarına göre daha az ayrıntılıdır.

Klasik Yaklaşım: V Model

- Test odaklı
- Şelale modelinin daha gelişmiş hali gibi düşünülebilir.
- Geliştirmedeki her doğrulama evresine karşın, testte bir onaylama evresi vardır.
- Döngü yoktur!
- Her adım için dokümasyon.



- **Çevik yazılım açısından eleştiriler:**
  - Yazılım geliştirme sürecinden çok proje yöneticilerinin, muhasebecilerin ve avukatların işini kolaylaştırır.
  - Esnek değil, doğrusal ve değişikliklere yanıt vermez.

Kaynaklar

- What is the Difference Between a Test Case and a Test Scenario? https://uilicious.com/blog/difference-between-test-case-test-scenario/
- Wikipedia – Test Driven Development https://en.wikipedia.org/wiki/Test-driven_development
- SDLC V-Model – Software Engineering https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/