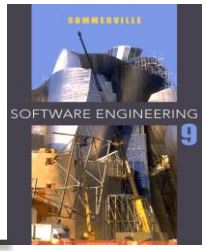
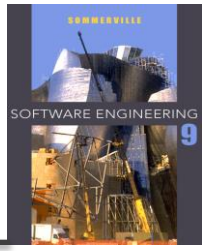


Chapter 8 – Yazılım Testi

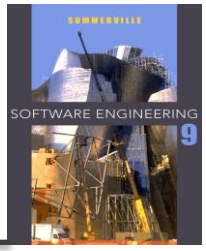
Lecture 1

Konular



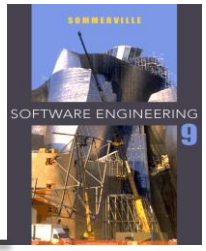
- Geliştirme testi
- Test tabanlı geliştirme
- Sürüm testi
- Kullanıcı testi

Programı test etmek



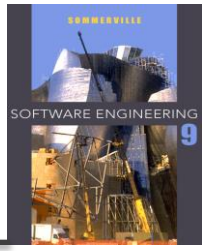
- Test etmek, bir programın yapması gereken şeyi yaptığını doğrulamak ve programdaki hataları keşfetmek için program kullanılmadan önce yapılan işlemdir.
- Bir yazılımı test ederken yapay veriler kullanılır.
- Çalıştırılan bir testin sonucunda, hatalar, anormallikler veya programın fonksiyonel olmayan özellikleri ile ilgili bilgiler edinilir.
- Test, Tetkik ve Tasdik sürecinin bir parçası olarak düşünülebilir.

Program testinin amaçları



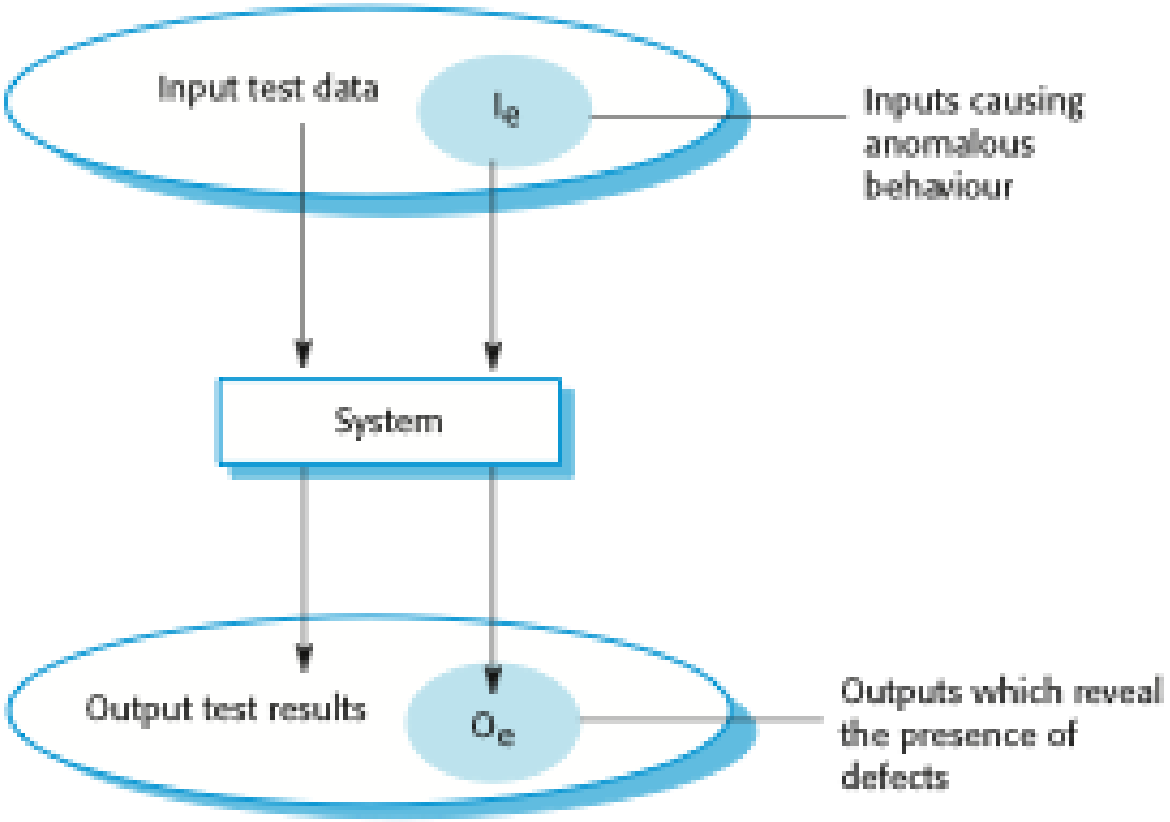
- Gösterim amaçlı testlerde yazılımın ihtiyaçları karşılayıp karşılamadığı gösterilir.
  - Özel amaçlı bir yazılımın testi için gereksinim dokümanındaki her bir gereksinim için en az bir test bulunmalıdır.
  - Genel amaçlı bir yazılımın testi için sistemin özelliklerini sınayacak testlerin yanı sıra bu özelliklerin birleşimi için de testler bulunmalıdır.
- Keşif amaçlı testlerde yazılımın hangi davranışlarının hatalı, istenmeyen olduğu veya gereksinimleri karşılamadığı gösterilir.
  - Kusur testi sistemin çökmeler, diğer sistemlerle istenmeyen etkileşimleri, yanlış hesaplamalar ve veri bozma gibi davranışlarına odaklıdır.

Doğrulama ve Kusur Testi

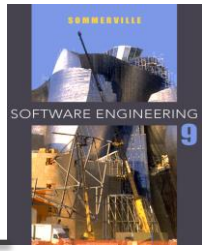


- Önceki sayfadaki ilk amaç doğrulama testini yönetir
  - Verilen bir test durumları kümesi için sistemin istenen çıktılar üretmesini bekleriz.
- Önceki sayfadaki ikinci amaç kusur testini yönetir
  - Verilen test durumları kümesi için sistemin hata verip vermediği gözlemlenir.

Program testi için bir giriş-çıkış modeli

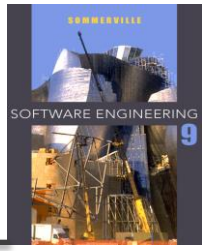


Verification (Tetkik) vs Validation (Tasdik)



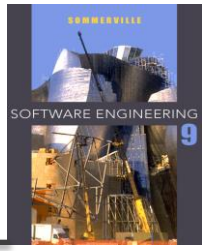
- Tetkik:
  «Ürünü doğru geliştiriyor muyuz?”.
- Yazılım, gereksinimleri sağlamalı
- Tasdik:
  «Doğru ürünü geliştiriyor muyuz?”.
- Yazılım gerçekte kullanıcının istediklerini sağlamalı

İnceleme ve Test



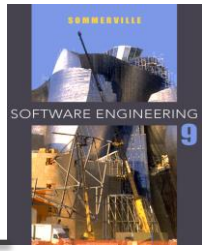
- Yazılım inceleme, statik olarak yapılan tetkiktir.
  - Dokümantasyon ve kod analizi şeklinde olabilir.
- Yazılım testi, dinamik olarak yapılan tetkiktir.
  - Sistem, test verisi ile çalıştırılır ve davranışları gözlemlenir.

Yazılım İncelemeleri



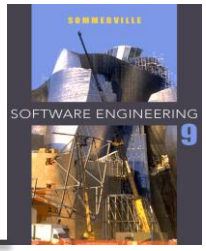
- İnsanların, kaynak koddaki hataları ve anormallikleri aramalarını kapsar.
- İnceleme, yazılımın çalıştırılmasını gerektirmediği için geliştirme sürecinden önce de yapılabilir.
- Sistemin herhangi bir «gösterimi» üzerinde yapılabilir. (gereksinimler, tasarım, konfigürasyon verisi, test verisi.)
- Program hatalarının keşfi için etkili bir yöntemdir.

İncelemenin avantajları



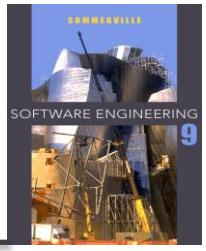
- Test esnasında bazı hatalar diğer hataları maskeleyebilir. İnceleme statik bir süreç olduğu için hataların etkileşimleri ile ilgilenmenize gerek yoktur.
- Sistemin henüz bitmemiş sürümleri incelenebilir.
- Programdaki hataları arama şeklinde olabileceği gibi aynı zamanda istenen standartları sağlayıp sağlamadığı, taşınılabilirlik ve sürdürülebilirlik gibi özellikler de incelenebilir.

İnceleme ve Test



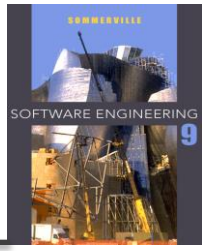
- İnceleme ve Test birbirlerini tamamlayan şeylerdir. Biri diğerinin yerine kullanılamaz.
- İnceleme bir gereksinimin sağlandığını gösterebilir ancak kullanıcının gerçek ihtiyacını onaylayamaz.
- İnceleme, performans, kullanılabilirlik gibi fonksiyonel olmayan gereksinimlerin kontrol edilmesinde kullanılmaz.

Testin aşamaları



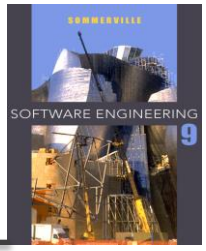
- Geliştirme testi, sistemin geliştirilmesi aşamasındaki hataların yakalanması.
- Sürüm testi, farklı bir test ekibi tarafından sistemin sürümden önce test edilmesi.
- Kullanıcı testi, kullanıcıların sistemi kendi ortamlarında test etmeleri.

Geliştirme testi



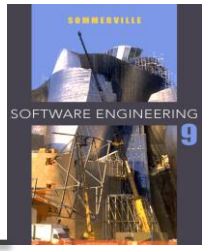
- Geliştirme testinin bütün aşamaları geliştirme ekibi tarafından yapılır.
  - Birim testi, programın birimlerinin veya nesne sınıflarının test edilmesidir. Bu birimlerin fonksiyonellikleri ile ilgilenilir.
  - Bileşen testi, bireysel bileşenlerin birleştirilerek daha büyük bileşenler oluşturulması aşamasında uygulanır. Bileşen testi, bileşenlerin arayüzlerine odaklanmalıdır.
  - Sistem testi, sistemin bütün olarak test edildiği aşamadır. Sistem testi, bileşenlerin etkileşimlerine odaklanmalıdır.

Birim testi



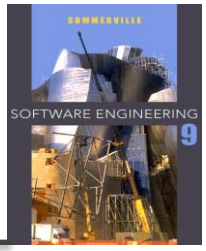
- ✧ Bireysel bileşenlerin «izole» edilerek test edilmesi.
- ✧ Hata bulmayı amaçlar
- ✧ Olası bileşenler şunlardır:
  - Bir nesnedeki fonksiyonlar veya metodlar
  - Birçok özellik veya metoddan oluşan nesne sınıfları
  - Birden çok arayüze sahip kompozit bileşenler.

Otomatikleştirilmiş test

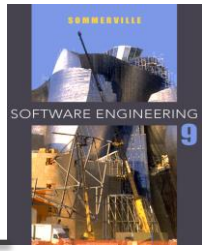


- Eğer mümkünse, birim testi el değmeden otomatikleştirilmiş olarak yapılmalıdır.
- Otomatikleştirilmiş birim testi için test otomasyon frameworkleri kullanarak testler yazıp çalıştırabilirsiniz.
- Birim testi otomasyon frameworkleri genel amaçlı test sınıfları sağlarlar.

Otomatikleştirilmiş testin bileşenleri



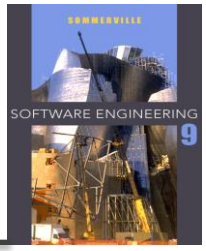
- Kurma aşaması, girişlerin ve istenen çıkışların belirlendiği aşama.
- Çağırma aşaması, test edilecek nesnenin «çağırıldığı» aşama.
- Değerlendirme aşaması, sonuçların istenen spnuçlarla karşılaştırıldığı aşama.



Chapter 8 – Yazılım Testi

Lecture 2

Bileşen Testi

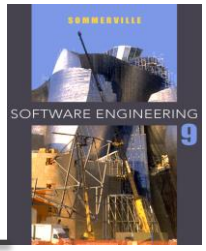


- Yazılım bileşenleri genelleikle birden fazla nesnenin etkileşimi ile oluşan kompozit bileşenlerdir.

- Bu bileşenlerin fonksiyonelliklerine arayüzleri üzerinden ulaşılır.

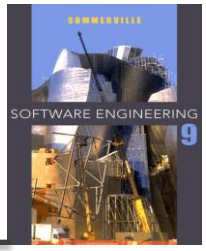
- Bundan dolayı bileşen testi, ilgili bileşenin aryüzünün nasıl davrandığına odaklanır.
  - Bu aşamada, birim testlerinin yapılmış olduğu varsayılabilir.

Arayüz testi



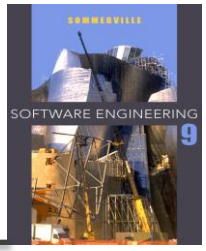
- ✧ Amaç, arayüzlerdeki hataları yakalamaktır.
- ✧ Arayüz tipler
  - Parametre arayüzleri veri bir metoddan diğerine aktarılır.
  - Paylaşılan hafıza aryüzleri bir hafıza bloğu metodlar arasında paylaşılır.
  - Prosedürel arayüzler Alt sistem, başka alt sistemler tarafından çağırılabilecek fonksiyonları kapsüller.
  - Mesajlaşma arayüzleri Alt sistem, başka alt sistemlerden servis ister.

Arayüz hataları



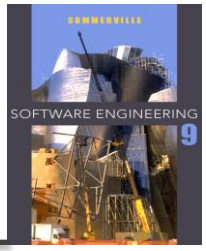
- ✧ Arayüzün hatalı kullanımı
  - Parametrelerin yanlış sırada verilmesi gibi.
- ✧ Arayüzün yanlış anlaşılması
  - Bir ikili arama fonksiyonunun sırasız dizi ile çağırılması
- ✧ Zamanlama hataları
  - Çağıran ve çağrılan metodların farklı hızlarda çalışması.

Arayüz testi için öneriler



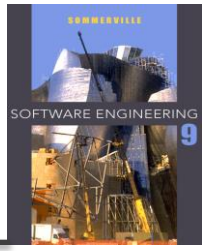
- ✧ Verilerin sınır değerlerini kullanarak test verisini hazırla.
- ✧ Pointer parametreleri mutlaka null pointer ile dene.
- ✧ Mesajlaşma sistemlerini stres testi ile test et.
- ✧ Paylaşılan bir hafızanın kullanılması durumunda bileşenlerin çalışma sıralarını değiştirerek dene.

Sistem testi



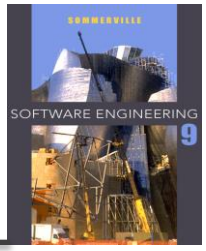
- Sistem testi bütün sistem üzerinde yapılır.
- Amacı, sistem bileşenlerinin etkileşimlerini test etmektir.
- Sistem testi bileşenlerin uyumluluğunu, düzgün etkileşimlerini ve doğru veriyi doğru zamanda arayüzlerinden aktarabildiklerini sınar.
- Sistem testi, sistemin yeni çıkan özelliklerini test eder.

Kulanım durumları ile test



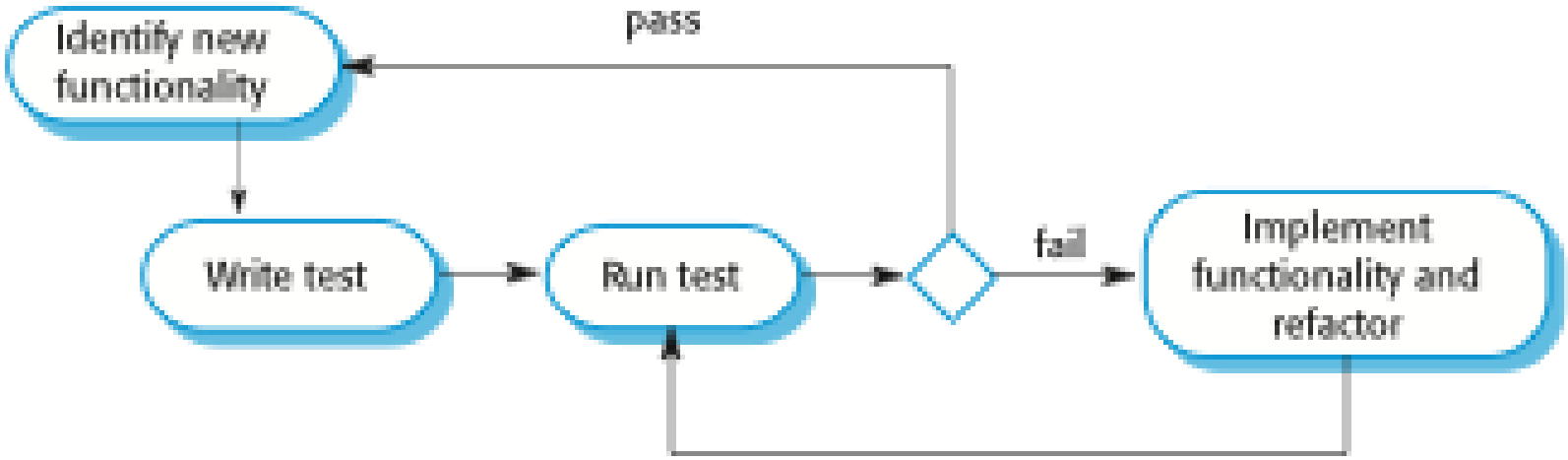
- Kullanım durumları birden fazla sistem bileşeninin etkileşimini içerir. Bu yüzden kullanım durumları, sistem testinde kullanılabilir.

Test tabanlı geliştirme (TTG)

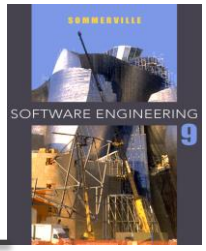


- TTG, geliştirme ile testin iç içe geçtiği bir durumdur.
- Testler kodlamadan önce geliştirilir ve bir bileşenin testi geçmesi geliştirme süreci için önemlidir.
- Kod, artırımlı olarak geliştirilir. Bir kod parçası testi geçemezse sonraki aşamaya geçilmez.
- TTG, çevik yöntemlerin bir parçası olarak ortaya çıkmıştır ancak plan tabanlı yöntemlerle de kullanılabilir.

Test tabanlı geliştirme (TTG)

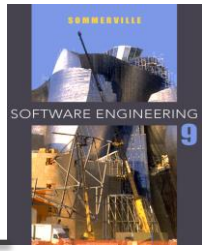


Test tabanlı geliştirme (TTG) süreç aktiviteleri



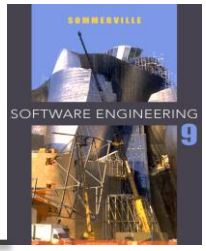
- Bir fonksiyonellik için bir artırım belirle.
- Bu artırım için test oluştur ve otomatik olarak geliştir.
- Testi, şimdiye kadar yazılmış diğer bütün testlerle birlikte sırası ile çalıştır.
- Fonksiyonelliği geliştir ve testi yeniden çalıştır.
- Bütün testler başarılı olduklarında sonraki fonksiyonelliğe geç

Test tabanlı geliştirmenin faydaları



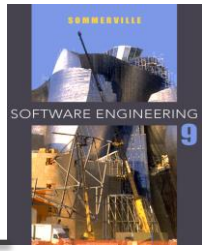
- Kodların kapsanması
  - Bütün kod parçaları en az bir kez test edilmiş olur.
- Regresyon testi
  - Program geliştirildikçe regresyon test yapısı da geliştirilmiş olur.
- Basitleştirilmiş hata ayıklama
  - Bir test hata ile sonuçlanınca hatayı nerde arayacağınızı bilirsiniz.
- Sistem dokümantasyonu
  - Testler aynı zamanda birer dokümandır ve hangi bileşenin ne yapması gerektiğini söyler.

Sürüm testi



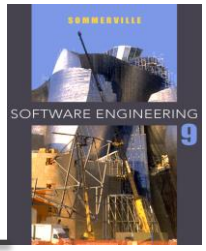
- Sürüm testi, bir sistemin belirli bir sürümünün geliştirme ekibi haricindeki kişilerce test edilmesidir.
- Bu testin birincil amacı, yazılımın kullanılacak durumda olup olmadığının sınanmasıdır.
  - Bundan dolayı sürüm testi bir sistemin yapması gerekenleri yaptığını, performansını ve güvenilirliğini gösterir. Ayrıca, normal kullanım durumunda hata vermediğini gösterir.
- Sürüm testi «kara kutu» testi olarak da adlandırılır.

Sürüm testi ve Sistem testi



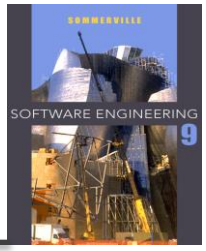
- Sürüm testi sistem testinin bir biçimidir.
- Önemli farkları:
  - Sürüm testini, geliştirme ekibinin haricindeki bir ekip yapar.
  - Geliştirme ekibi tarafından yapılan sistem testi, sistemin hatalarını bulmaya odaklıdır.
  - Sürüm testinin amacı, sistemin gereksinimleri karşılayıp karşılamadığının belirlenmesidir. Böylece, sistemin kullanıma hazır olup olmadığı sorusunun cevabı bulunabilir.

Performansı test etmek



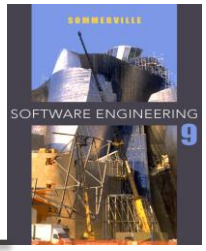
- Sürüm testinin bir parçası da performansı test etmektir.
- Testler, sistemin kullanım profiline uygun olmalıdır.
- Performans testleri genellikle sistemi artan biçimde yükleyen testler serisi olarak planlanır ve sistemin performansının «kabul edilemez» olarak adlandırıldığı noktaya kadar yapılır.
- Stres testi, performans testinin bir biçimidir.

Kullanıcı testi



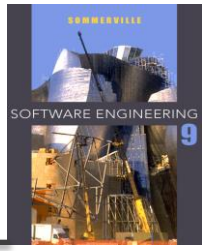
- Kullanıcı testi (müşteri testi), girişlerin kullanıcılar tarafından verildiği ve sistemin değerlendirildiği bir aşamadır.
- Kapsamlı bir sistem ve sürüm testi yapılmış olsa bile kullanıcı testi gereklidir.
  - Bunun nedeni, kullanıcının ortamının sistemin çalışması üzerinde büyük etkisinin olmasıdır. Bu durum bir test ortamında oluşturulabilecek bir durum değildir.

Kullanıcı test tipleri



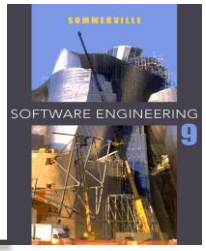
- Alpha testi
  - Geliştirme ekibi ile beraber çalışan kullanıcılar sistemi geliştirilen ortamda test eder.
- Beta testi
  - Sistemin bir sürümü kullanıcılara açılır. Kullanıcılar sistem üzerindeki deneyimlerini ve çıkan problemleri geliştiriciler ile paylaşır.
- Kabul testi
  - Müşteriler sistemin gerçek çalışma ortamında aktarılabilecek durumda olup olmadığını test ederler.

Kabul testi sürecinin aşamaları



- ✧ Kabul kriterlerini tanımla
- ✧ Kabul testini planla
- ✧ Kabul testini geliştir
- ✧ Kabul testini çalıştır
- ✧ Test sonuçlarını değerlendir
- ✧ Sistemi kabul et veya reddet

Çevik yöntemler ve kabul testi



- Çevik yaklaşımlarda müşteri geliştirme ekibinin bir parçasıdır ve sistemin kabul edilmesi üzerine karar verebilir.
- Testler müşteri tarafından tanımlanır.
- Ayrı bir kabul testi yoktur.
- Buradaki sorun, geliştirme ekibine dahil edilen kullanıcının «paydaşları» ne kadar temsil ettiğidir.