IT522 – Yazılım Mühendisliği 2021

PhD Furkan Gözükara, Toros University

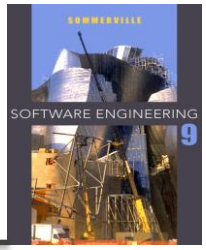
https://github.com/FurkanGozukara/Yazilim-Muhendisligi-IT522-2021

Ders 8

Yazılım Testi

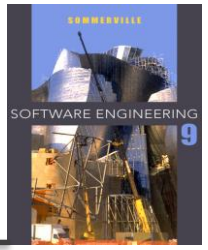
Kaynak : https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Presentations/index.html

İşlenmiş Konular



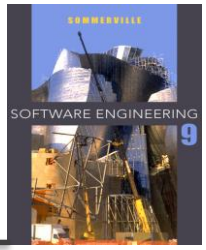
- Geliştirme testi
- Test odaklı geliştirme
- Yayın testi
- Kullanıcı testi

Program Testi



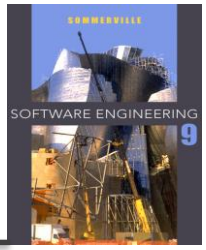
- Test, bir programın yapması amaçlanan şeyi yaptığını göstermeyi ve program kullanılmadan önce hatalarını keşfetmeyi amaçlamaktadır.
- Yazılımı test ettiğinizde, yapay verileri kullanarak bir program yürütürsünüz.
- Programın işlevsel olmayan öznitelikleri hakkında hatalar, anormallikler veya bilgiler için test çalıştırılmasının sonuçlarını kontrol edersiniz.
- Hataların yokluğunu DEĞİL, varlığını ortaya çıkarabilir.
- Test, statik doğrulama tekniklerini de içeren daha genel bir doğrulama ve onaylama sürecinin bir parçasıdır.

Program Test Hedefleri



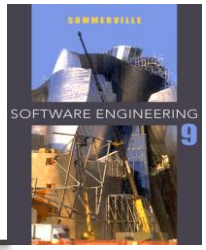
- Yazılımın gereksinimlerini karşıladığını geliştiriciye ve müşteriye göstermek.
  - Özel yazılım için bu, gereksinimler belgesindeki her gereksinim için en az bir test olması gerektiği anlamına gelir. Genel yazılım ürünleri için bu, ürün sürümüne dahil edilecek tüm sistem özellikleri ve bu özelliklerin kombinasyonları için testler yapılması gerektiği anlamına gelir.
- Yazılımın davranışının yanlış, istenmeyen veya spesifikasyonuna uymadığı durumları keşfetmek.
  - Hata testi, sistem çökmeleri, diğer sistemlerle istenmeyen etkileşimler, yanlış hesaplamalar ve veri bozulması gibi istenmeyen sistem davranışlarını ortadan kaldırmakla ilgilidir.

Doğrulama ve Kusur Testi



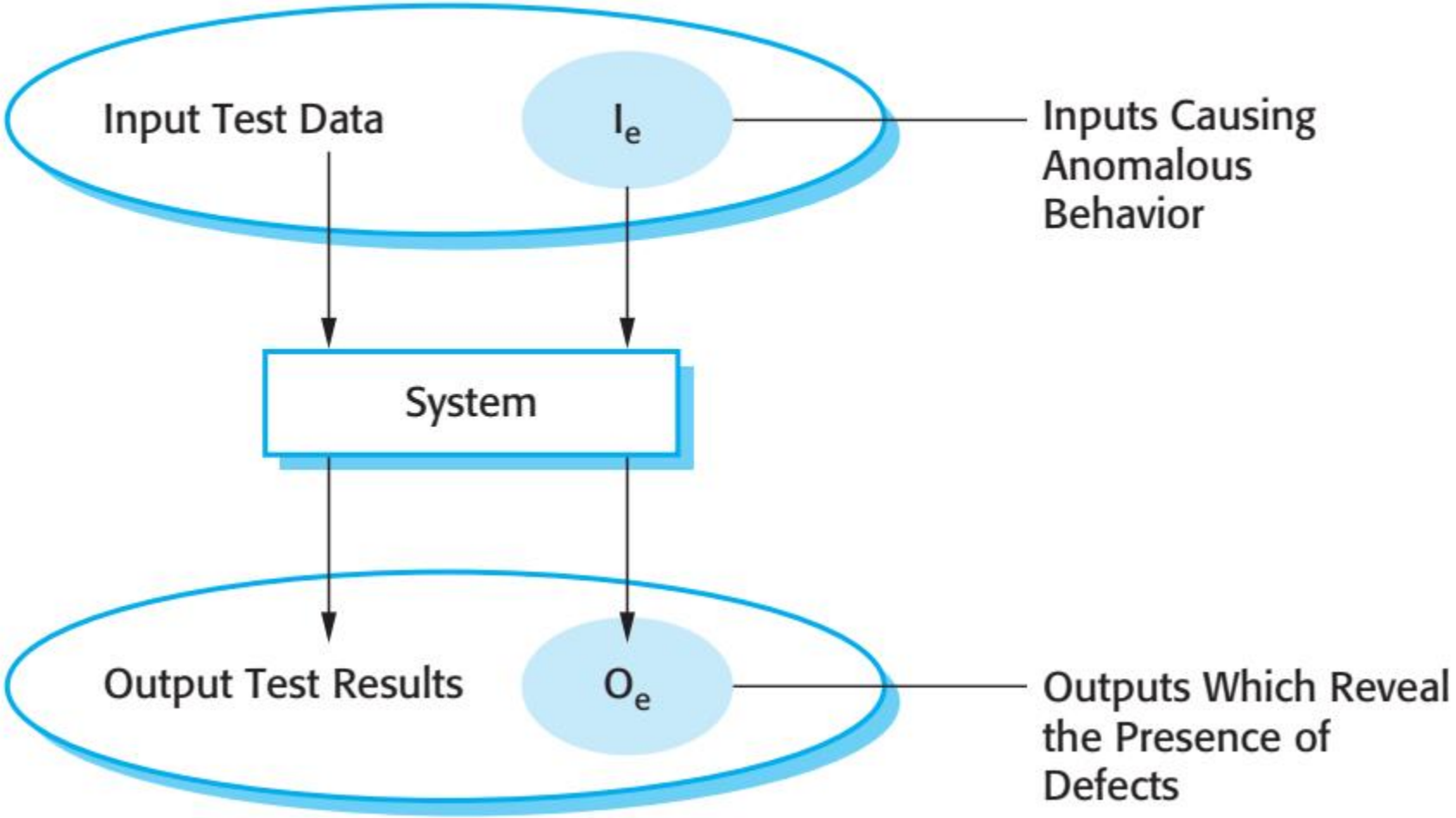
- İlk hedef doğrulama testine götürür
  - Sistemin beklenen kullanımını yansıtan belirli bir test senaryosu seti kullanarak doğru şekilde çalışmasını beklersiniz.
- İkinci hedef, kusur testine götürür
  - Test senaryoları, kusurları ortaya çıkarmak için tasarlanmıştır. Hata testindeki test senaryoları kasıtlı olarak belirsiz olabilir ve sistemin normal olarak nasıl kullanıldığını yansıtması gerekmez.

Test Süreci Hedefleri

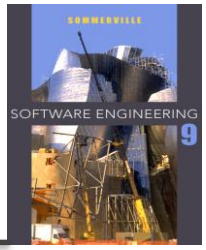


- Doğrulama testi
  - Geliştiriciye ve sistem müşterisine yazılımın gereksinimlerini karşıladığını göstermek için
  - Başarılı bir test, sistemin amaçlandığı gibi çalıştığını gösterir.
- Hata testi
  - Yazılımın davranışının yanlış olduğu veya spesifikasyonuna uygun olmayan arızaları veya kusurları keşfetmek için
  - Başarılı bir test, sistemin hatalı çalışmasına neden olan ve bu nedenle sistemdeki bir kusuru ortaya çıkaran bir testtir.

Program Testinin Bir Girdi-Çıktı Modeli

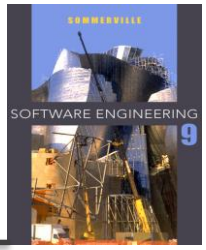


Doğrulama (Validation) ve Sağlama (Verification)



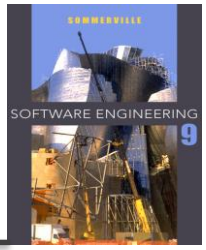
- Doğrulama: "Ürünü doğru geliştiriyor muyuz".
  - Yazılım, teknik özelliklerine uygun olmalıdır.
- Sağlama: "Doğru ürünü mü geliştiriyoruz".
  - Yazılım, kullanıcının gerçekten ihtiyaç duyduğu şeyi yapmalıdır.

Doğrulama ve Sağlama



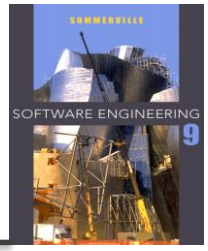
- Doğrulama ve Sağlama'nın amacı, sistemin "amaca uygun" olduğuna dair güven oluşturmaktır.
- Sistemin amacına, kullanıcı beklentilerine ve pazarlama ortamına bağlıdır
  - Yazılım amacı
    - Güven seviyesi, yazılımın bir kuruluş için ne kadar kritik olduğuna bağlıdır.
  - Kullanıcı beklentileri
    - Kullanıcıların belirli yazılım türlerinden beklentileri düşük olabilir.
  - Pazarlama ortamı
    - Bir ürünü pazara erken sürmek, programdaki kusurları bulmaktan daha önemli olabilir.

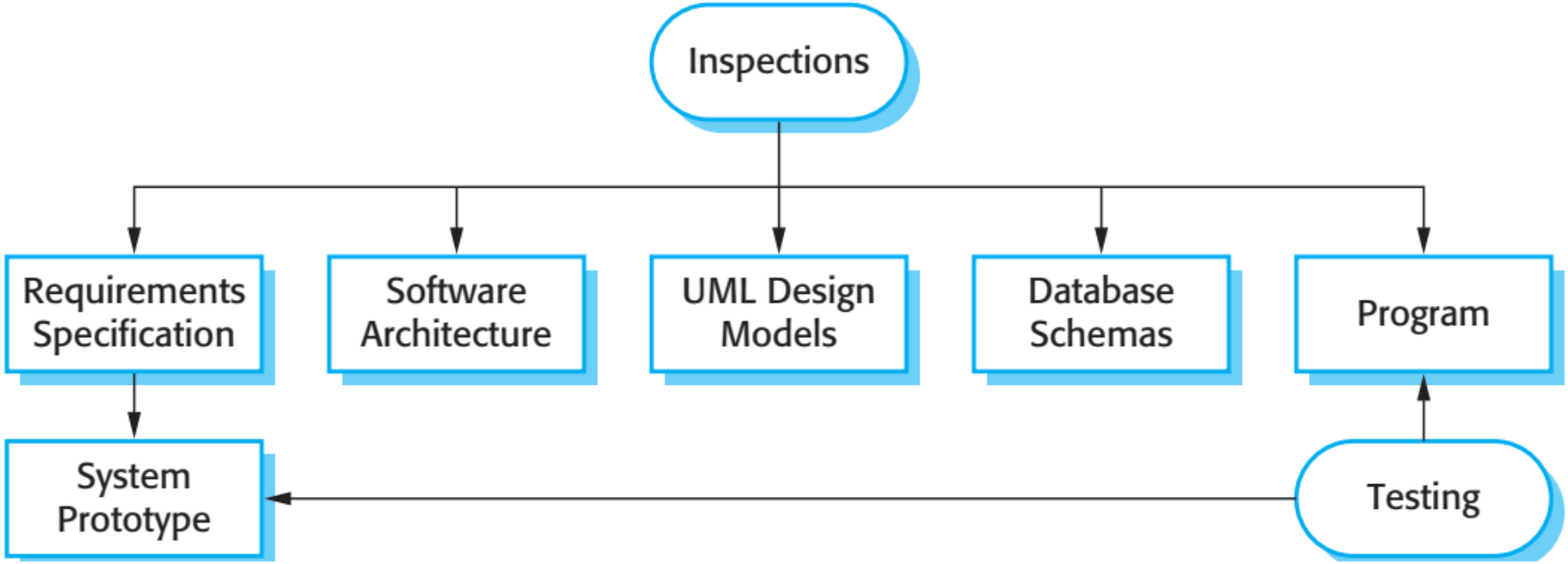
İncelemeler ve Testler



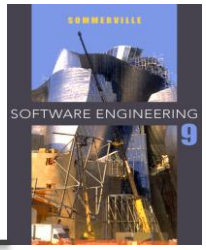
- Yazılım incelemeleri Sorunları keşfetmek için statik sistem temsilinin analizi ile ilgili (statik doğrulama)
  - Araç tabanlı belge ve kod analizi ile tamamlanabilir.
  - Ders 15'te tartışıldı.
- Yazılım testi Ürün davranışını uygulama ve gözlemleme ile ilgili (dinamik doğrulama)
  - Sistem test verileri ile çalıştırılır ve operasyonel davranışı gözlemlenir.

İncelemeler ve Testler



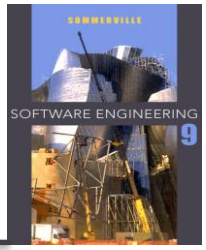


Yazılım İncelemeleri



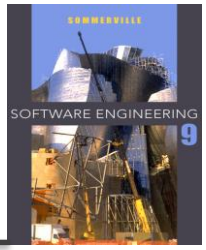
- Bunlar, anormallikleri ve kusurları keşfetmek amacıyla kaynak temsilini inceleyen insanları içerir.
- Denetimler, bir sistemin yürütülmesini gerektirmediğinden, uygulamadan önce kullanılabilir.
- Sistemin herhangi bir temsiline uygulanabilir (gereksinimler, tasarım, konfigürasyon verileri, test verileri, vb.).
- Program hatalarını keşfetmek için etkili bir teknik oldukları gösterilmiştir.

İncelemelerin Avantajları



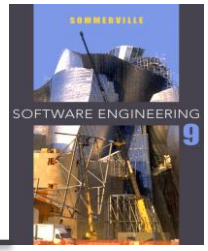
- Test sırasında, hatalar diğer hataları maskeleyebilir (gizleyebilir). İnceleme statik bir süreç olduğundan, hatalar arasındaki etkileşimlerle ilgilenmenize gerek yoktur.
- Bir sistemin eksik sürümleri ek maliyet olmadan incelenebilir. Bir program eksikse, mevcut parçaları test etmek için özel test donanımları geliştirmeniz gerekir.
- Bir inceleme, program hatalarını aramanın yanı sıra, standartlara uyum, taşınabilirlik ve sürdürülebilirlik gibi bir programın daha geniş kalite özelliklerini de dikkate alabilir.

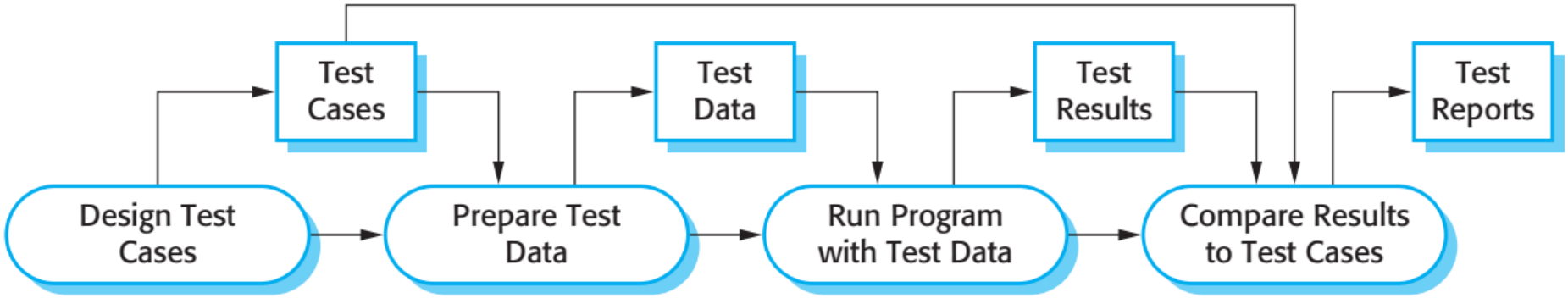
İncelemeler ve Testler



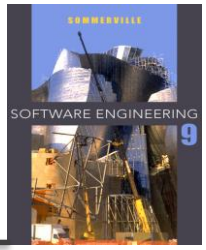
- İncelemeler ve testler birbirini tamamlayıcı niteliktedir ve doğrulama tekniklerine ters düşmez.
- Her ikisi de Doğrulama ve Sağlama işlemi sırasında kullanılmalıdır.
- İncelemeler, bir spesifikasyona uygunluğu kontrol edebilir, ancak müşterinin gerçek gereksinimlerine uygunluğu kontrol edemez.
- İncelemeler, performans, kullanılabilirlik vb. gibi işlevsel olmayan özellikleri kontrol edemez.

Yazılım Test Sürecinin Bir Modeli



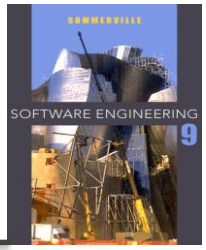


Test Aşamaları



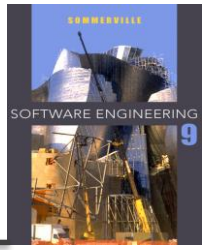
- Geliştirme sırasında sistemin hatalarını ve kusurlarını keşfetmek için test edildiği geliştirme testi.
- Ayrı bir test ekibinin, kullanıcılara sunulmadan önce sistemin eksiksiz bir sürümünü test ettiği sürüm testi.
- Kullanıcıların veya bir sistemin potansiyel kullanıcılarının sistemi kendi ortamlarında test ettiği kullanıcı testi.

Geliştirme Testi



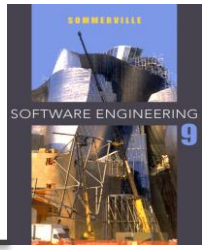
- Geliştirme testleri, sistemi geliştiren ekip tarafından gerçekleştirilen tüm test faaliyetlerini içerir.
  - Ayrı program birimlerinin veya nesne sınıflarının test edildiği birim testi. Birim testi, nesnelerin veya yöntemlerin işlevselliğini test etmeye odaklanmalıdır.
  - Bileşik bileşenler oluşturmak için birkaç bağımsız birimin entegre edildiği bileşen testi. Bileşen testi, bileşen arayüzlerini test etmeye odaklanmalıdır.
  - Bir sistemdeki bileşenlerin bir kısmının veya tümünün entegre edildiği ve sistemin bir bütün olarak test edildiği sistem testi. Sistem testi, bileşen etkileşimlerini test etmeye odaklanmalıdır.

Birim Testi



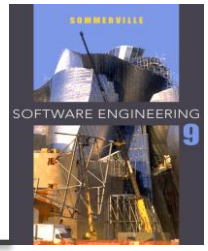
- Birim testi, tek tek bileşenlerin ayrı ayrı test edilmesi sürecidir.
- Hata testi sürecidir.
- Birimler şunlar olabilir:
  - Bir nesne içindeki bireysel işlevler veya yöntemler
  - Çeşitli niteliklere ve yöntemlere sahip nesne sınıfları
  - İşlevselliklerine erişmek için kullanılan tanımlı arayüzlere sahip kompozit bileşenler.

Nesne Sınıfı Testi



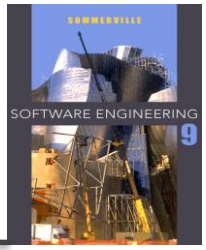
- Bir sınıfın eksiksiz test kapsamı şunları içerir:
  - Bir nesneyle ilişkili tüm işlemleri test etme
  - Tüm nesne niteliklerini ayarlama ve sorgulama
  - Nesneyi tüm olası durumlarda uygulamak.
- Kalıtım, test edilecek bilgiler yerelleştirilmediğinden nesne sınıfı testlerinin tasarlanmasını zorlaştırır.

Meteoroloji İstasyonu Nesne Arayüzü



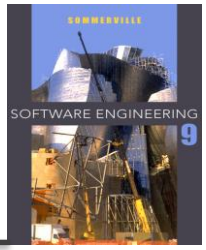
WeatherStation
identifier
reportWeather ( ) reportStatus ( ) powerSave (instruments) remoteControl (commands) reconfigure (commands) restart (instruments) shutdown (instruments)

Hava İstasyonu Testi



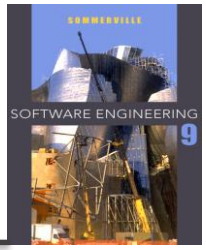
- ReportWeather, kalibrasyon, test, başlatma ve kapatma için test senaryoları tanımlamanız gerekir.
- Bir durum modeli kullanarak, test edilecek durum geçişleri dizilerini ve bu geçişlere neden olacak olay dizilerini tanımlayın.
- Örneğin:
  - Kapatma -> Çalışıyor-> Kapatma
  - Yapılandırma-> Çalıştırma-> Test -> Gönderme -> Çalışıyor
  - Çalıştırma-> Toplama-> Çalıştırma-> Özetleme -> İletme -> Çalıştırma

Otomatik Test



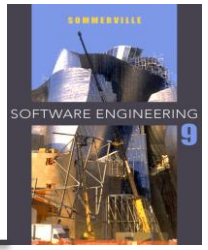
- Mümkün olduğunda, birim testi otomatikleştirilmelidir, böylece testler manuel müdahale olmadan çalıştırılır ve kontrol edilir.
- Otomatik birim testinde, program testlerinizi yazmak ve çalıştırmak için bir test otomasyon çerçevesi(framework) (JUnit gibi) kullanırsınız.
- Birim testi çerçeveleri, belirli test senaryoları oluşturmak için genişlettiğiniz genel test sınıfları sağlar. Daha sonra uyguladığınız tüm testleri çalıştırabilir ve genellikle bazı GUI'ler (grafiksel kullanıcı arayüzleri) aracılığıyla testlerin aksi yöndeki başarısını rapor edebilirler.

Otomatik Test Bileşenleri



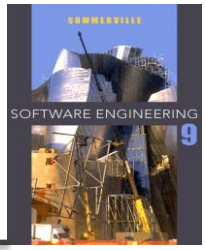
- Sistemi test senaryosu ile başlattığınız bir kurulum bölümü, yani girişler ve beklenen çıkışlar.
- Test edilecek nesneyi veya yöntemi çağırdığınız bir çağrı bölümü.
- Çağrının sonucunu beklenen sonuçla karşılaştırdığınız bir onaylama bölümü. İddia doğru olarak değerlendirilirse, test başarılı olmuştur, eğer doğru değilse, test başarısız olmuştur.
- JUnit ile otomatik test> https://www.youtube.com/watch?v=I8XXfgF9GSc

Birim Test Etkinliği



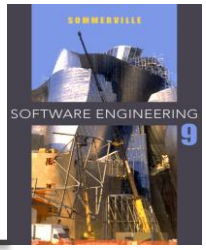
- Test senaryoları, beklendiği gibi kullanıldığında, test ettiğiniz bileşenin yapması gerekeni yaptığını göstermelidir.
- Bileşende kusurlar varsa, bunlar test senaryoları ile ortaya çıkarılmalıdır.
- Bu, 2 tür birim test senaryosuna yol açar:
  - Bunlardan ilki, bir programın normal işleyişini yansıtmalı ve bileşenin beklendiği gibi çalıştığını göstermelidir.
  - Diğer tür test senaryosu ise, yaygın sorunların ortaya çıktığı yerdeki test deneyimine dayanmalıdır. Bunların düzgün şekilde işlendiğini ve bileşeni çökertmediğini kontrol etmek için anormal girdiler kullanmalıdır.

Test Stratejileri



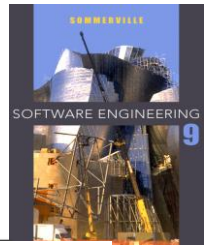
- Ortak özelliklere sahip olan ve aynı şekilde işlenmesi gereken girdi gruplarını belirlediğiniz bölümleme testi.
  - Bu grupların her biri içinden test seçmelisiniz.
- Test senaryolarını seçmek için test yönergelerini kullandığınız kılavuza dayalı test.
  - Bu yönergeler, programcıların bileşenleri geliştirirken sıklıkla yaptıkları hata türlerine ilişkin önceki deneyimlerini yansıtır.

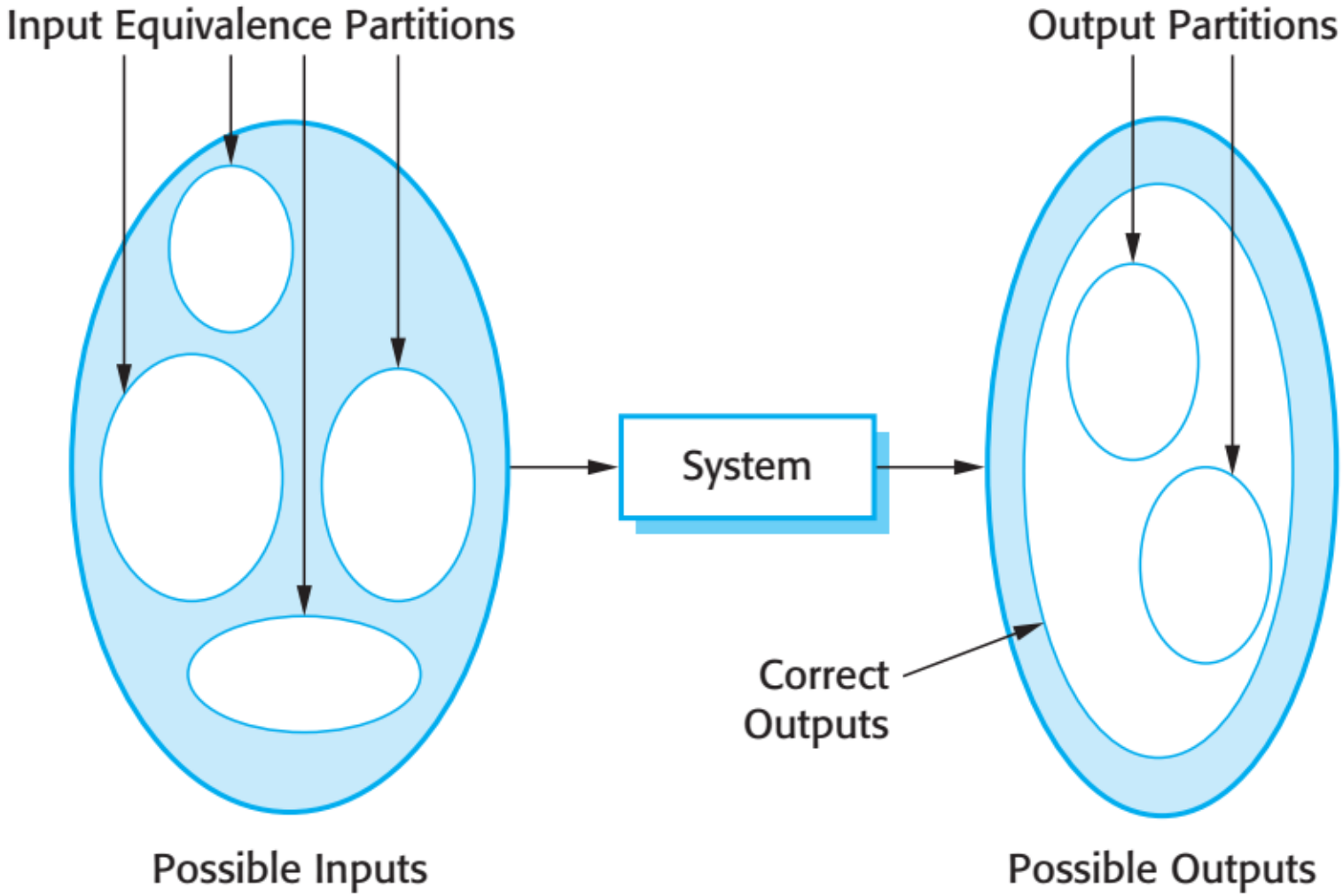
Bölme Testi



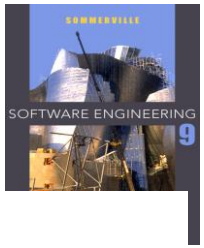
- Girdi verileri ve çıktı sonuçları genellikle bir sınıfın tüm üyelerinin ilişkili olduğu farklı sınıflara girer.
- Bu sınıfların her biri, programın her sınıf üyesi için eşdeğer bir şekilde davrandığı bir eşdeğerlik bölümü veya etki alanıdır.
- Her bölümden test senaryoları seçilmelidir.

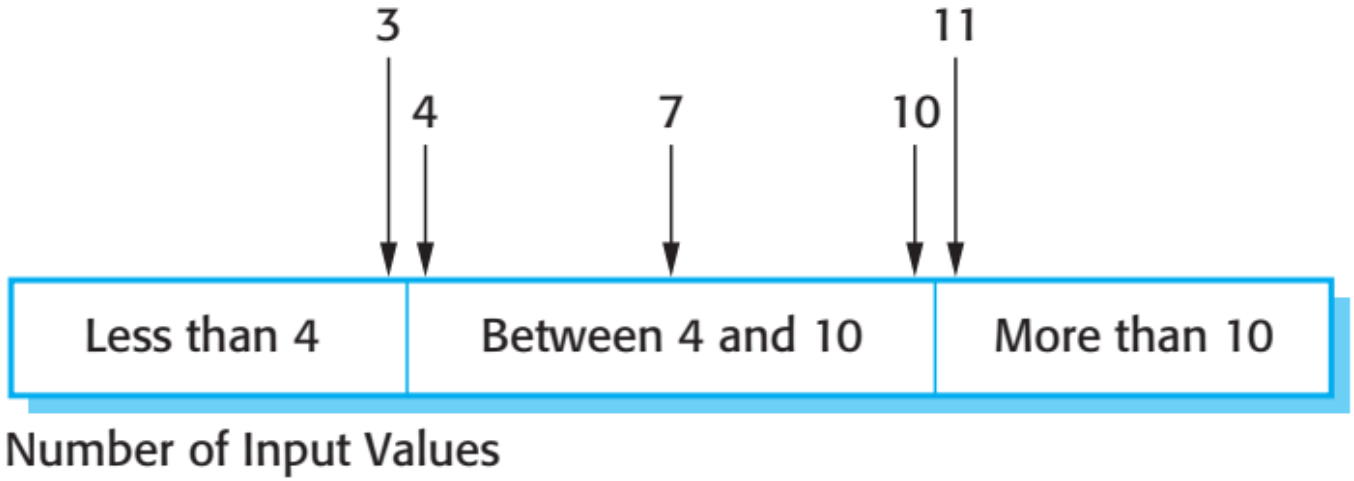
Eşit Bölümlere Ayırma

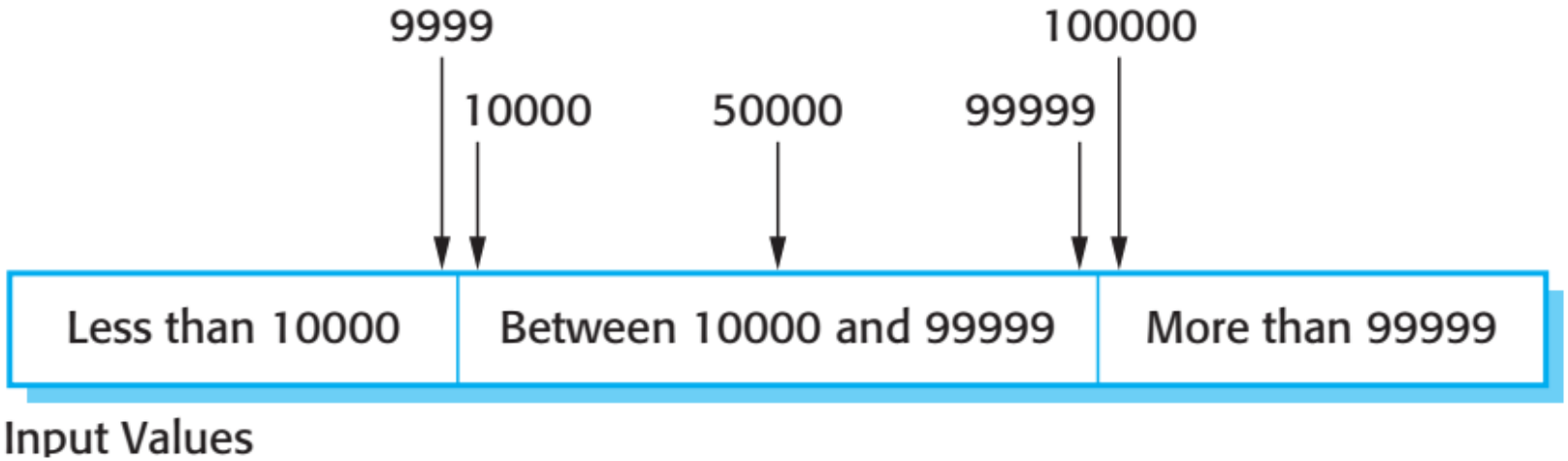




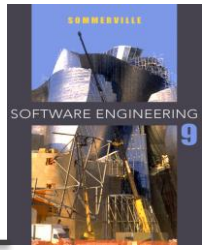
Eşdeğerlik Bölümleri





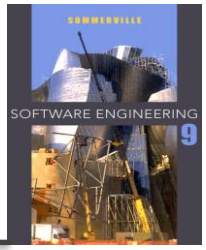


Test Yönergeleri (Diziler)



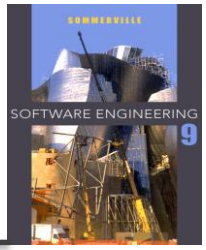
- Yalnızca tek bir değere sahip dizileri olan test yazılımı.
- Farklı testlerde farklı boyutlarda diziler kullanın.
- Dizinin ilk, orta ve son öğelerine erişilecek şekilde testleri türetin.
- Sıfır uzunluktaki dizilerle test edin.

Genel Test Yönergeleri

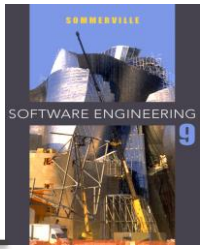


- Sistemi tüm hata mesajlarını oluşturmaya zorlayan girişleri seçin
- Giriş arabelleklerinin taşmasına neden olan tasarım girdileri
- Aynı girişi veya bir dizi girişi defalarca tekrarlayın
- Geçersiz çıktıların oluşturulmasını zorla
- Hesaplama sonuçlarını çok büyük veya çok küçük olmaya zorlayın.

Bölüm 1'in Anahtar Noktaları



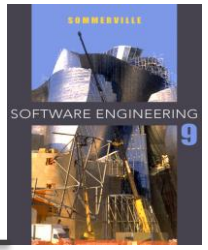
- Test, yalnızca bir programdaki hataların varlığını gösterebilir. Kalan hatanın olmadığını gösteremez.
- Geliştirme testi, yazılım geliştirme ekibinin sorumluluğundadır. Müşterilere sunulmadan önce bir sistemin test edilmesinden ayrı bir ekip sorumlu olmalıdır.
- Geliştirme testi, tek tek nesneleri test ettiğiniz birim testini ve ilgili nesne gruplarını test ettiğiniz bileşen testini ve kısmi veya tam sistemleri test ettiğiniz sistem testini içerir.



Ders 8 - Yazılım Testi

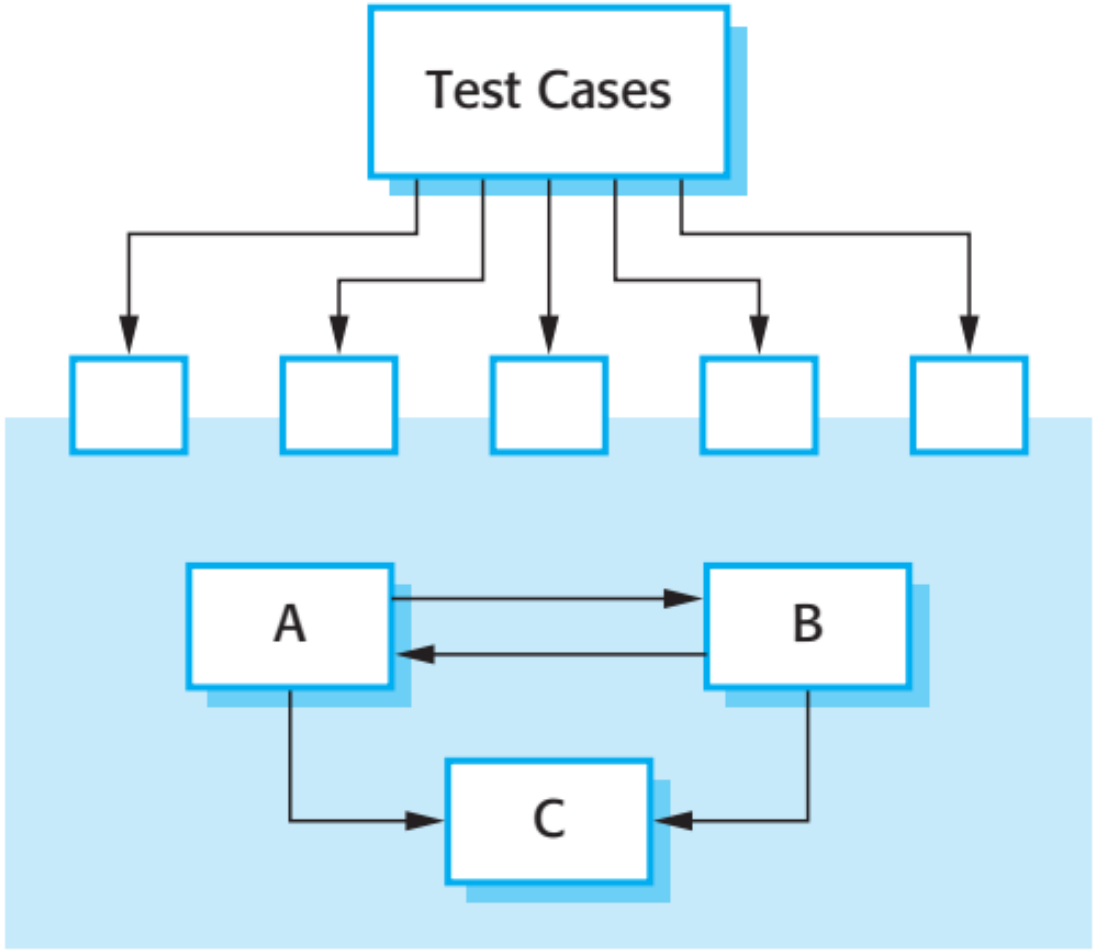
Bölüm 2

Bileşen Testi

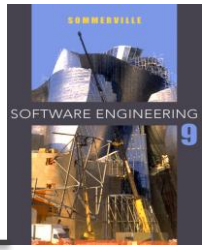


- Yazılım bileşenleri genellikle birkaç etkileşimli nesneden oluşan bileşik bileşenlerdir.
  - Örneğin, meteoroloji istasyonu sisteminde, yeniden yapılandırma bileşeni, yeniden yapılandırmanın her bir yönü ile ilgilenen nesneleri içerir.
- Bu nesnelerin işlevselliğine tanımlı bileşen arabirimi üzerinden erişirsiniz.
- Bu nedenle, kompozit bileşenlerin test edilmesi, bileşen arayüzünün özelliklerine göre davrandığını göstermeye odaklanmalıdır.
  - Bileşen içindeki tek tek nesneler üzerindeki birim testlerinin tamamlandığını varsayabilirsiniz.

Arayüz Testi

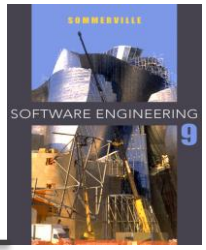


Arayüz Testi



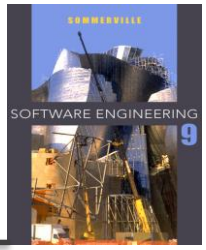
- Amaçlar, arayüz hataları veya arayüzler hakkında geçersiz varsayımlar nedeniyle oluşan hataları tespit etmektir.
- Arayüz türleri
  - Parametre arabirimleri Bir yöntem veya prosedürden diğerine geçen veriler.
  - Paylaşılan hafıza arayüzleri Hafıza bloğu prosedürler veya fonksiyonlar arasında paylaşılır.
  - Prosedürel arayüzler Alt sistem, diğer alt sistemler tarafından çağrılacak bir dizi prosedürü kapsüller.
  - Mesaj iletme arayüzleri Alt sistemler, diğer alt sistemlerden servis talep eder

Arayüz Hataları



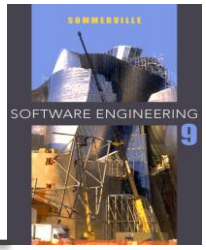
- Arayüzün yanlış kullanımı
  - Çağıran bir bileşen başka bir bileşeni çağırır ve arayüzünün kullanımında bir hata yapar, örneğin parametreler yanlış sırada.
- Arayüz yanlış anlaşılması
  - Çağıran bir bileşen, çağrılan bileşenin davranışı hakkında yanlış olan varsayımları uygular.
- Zamanlama hataları
  - Aranan ve arayan bileşen farklı hızlarda çalışır ve güncel olmayan bilgilere erişilir.

Arayüz Testi Yönergeleri



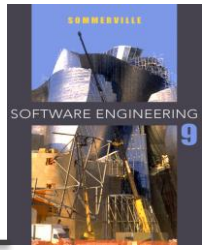
- Testleri, adlandırılan bir prosedürün parametrelerinin aralıklarının en uç noktalarında olması için tasarlayın.
- İşaretçi parametrelerini her zaman boş işaretçilerle test edin.
- Bileşenin başarısız olmasına neden olan tasarım testleri.
- Mesaj geçirme sistemlerinde stres testini kullanın.
- Paylaşılan bellek sistemlerinde, bileşenlerin etkinleştirilme sırasını değiştirin.

Sistem Testi



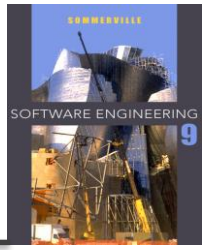
- Geliştirme sırasında sistem testi, sistemin bir sürümünü oluşturmak için bileşenleri entegre etmeyi ve ardından entegre sistemi test etmeyi içerir.
- Sistem testinin odak noktası, bileşenler arasındaki etkileşimleri test etmektir.
- Sistem testi, bileşenlerin uyumlu olup olmadığını, doğru etkileşimde bulunup bulunmadığını ve doğru verileri doğru zamanda arayüzleri üzerinden aktardıklarını kontrol eder.
- Sistem testi, bir sistemin ortaya çıkan davranışını test eder.

Sistem ve Bileşen Testi



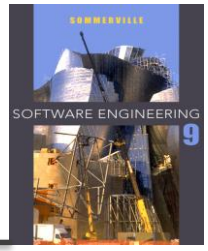
- Sistem testi sırasında, ayrı ayrı geliştirilen ve kullanıma hazır sistemlerdeki yeniden kullanılabilir bileşenler yeni geliştirilen bileşenlerle entegre edilebilir. Tüm sistem daha sonra test edilir.
- Farklı ekip üyeleri veya alt ekipler tarafından geliştirilen bileşenler bu aşamada entegre edilebilir. Sistem testi, bireysel bir süreçten ziyade toplu bir süreçtir.
  - Bazı şirketlerde, sistem testi, tasarımcıların ve programcıların katılımı olmaksızın ayrı bir test ekibini içerebilir.

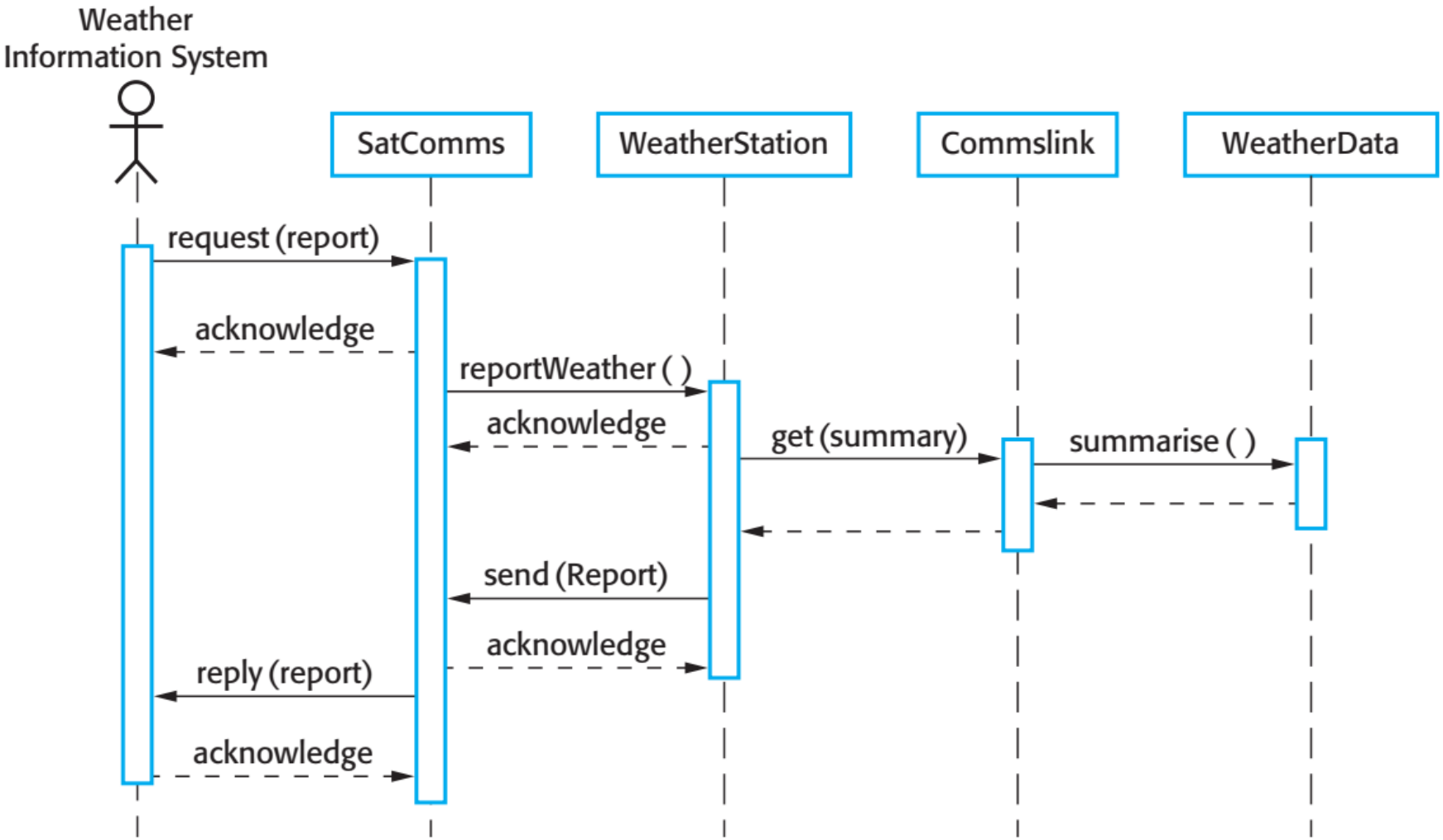
Kullanım Senaryosu Testi



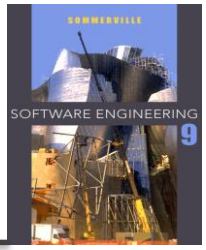
- Sistem etkileşimlerini tanımlamak için geliştirilen kullanım durumları, sistem testi için bir temel olarak kullanılabilir.
- Her kullanım senaryosu genellikle birkaç sistem bileşenini içerir, bu nedenle kullanım senaryosunun test edilmesi bu etkileşimleri oluşmaya zorlar.
- Kullanım senaryosuyla ilişkili sıra diyagramları, test edilen bileşenleri ve etkileşimleri belgeler.

Hava Durumu Verilerini Toplama Sıra Çizelgesi



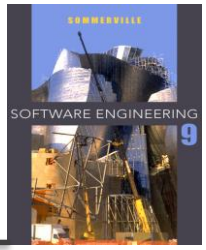


Test Politikaları



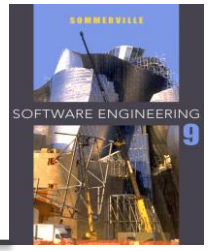
- Kapsamlı sistem testi imkansız olduğundan, gerekli sistem testi kapsamını tanımlayan test politikaları geliştirilebilir.
- Test politikalarına örnekler:
  - Menüler aracılığıyla erişilen tüm sistem işlevleri test edilmelidir.
  - Aynı menüden erişilen işlev kombinasyonları (örn. Metin biçimlendirme) test edilmelidir.
  - Kullanıcı girişi sağlandığında, tüm işlevler hem doğru hem de yanlış girişle test edilmelidir.

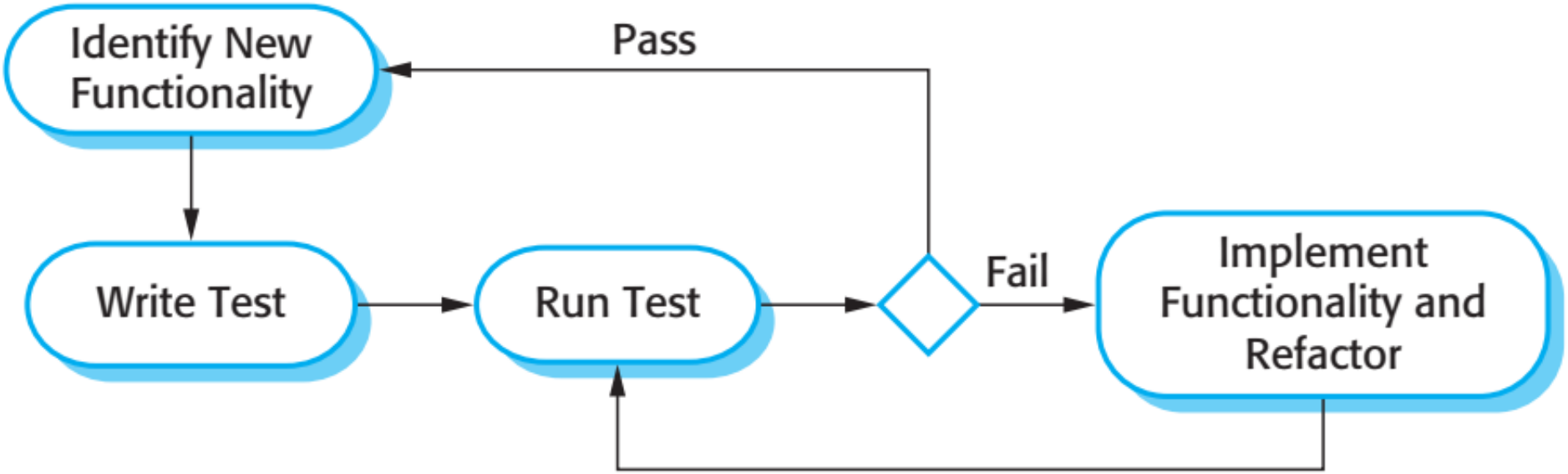
Test Odaklı Geliştirme



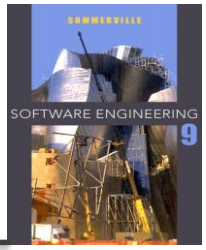
- Test güdümlü geliştirme (TGG), test ve kod geliştirmeyi aralıklı bıraktığınız program geliştirme yaklaşımıdır.
- Testler koddan önce yazılır ve testleri 'geçmek', geliştirmenin kritik itici gücüdür.
- Bu artım için bir test ile birlikte aşamalı olarak kod geliştirirsiniz. Geliştirdiğiniz kod testini geçene kadar bir sonraki aşamaya geçemezsiniz.
- TGG, Ekstrem Programlama gibi çevik yöntemlerin bir parçası olarak tanıtıldı. Ancak plan odaklı geliştirme süreçlerinde de kullanılabilir.

Test Odaklı Geliştirme



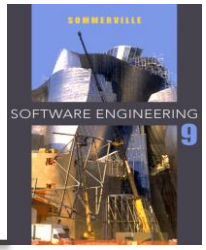


TGG Süreç Faaliyetleri



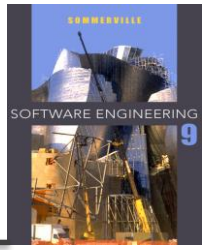
- Gerekli olan işlevsellik artışını belirleyerek başlayın. Bu normalde küçük ve birkaç satır kodda uygulanabilir olmalıdır.
- Bu işlevsellik için bir test yazın ve bunu otomatik bir test olarak uygulayın.
- Uygulanan diğer tüm testlerle birlikte testi çalıştırın. Başlangıçta, işlevselliği uygulamamışsınızdır, bu nedenle yeni test başarısız olacaktır.
- İşlevselliği programlayın ve testi yeniden çalıştırın.
- Tüm testler başarılı bir şekilde çalıştıktan sonra, bir sonraki işlevsellik parçasını uygulamaya/geliştirmeye geçersiniz.

Test Odaklı Geliştirmenin Faydaları



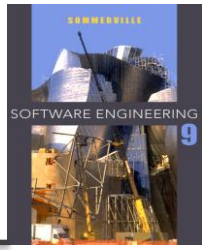
- Kod kapsamı
  - Yazdığınız her kod segmentinin en az bir ilişkili testi vardır, bu nedenle yazılan tüm kodun en az bir testi vardır.
- Gerileme testi
  - Bir program geliştirildikçe aşamalı olarak bir regresyon testi paketi geliştirilir. Acaba yeni özellikler ekledikten sonar mevcut sisteminizdeki herhangi bir kısım bozuldu mu?
- Basitleştirilmiş hata ayıklama
  - Bir test başarısız olduğunda, sorunun nerede olduğu belli olmalıdır. Yeni yazılan kodun kontrol edilmesi ve değiştirilmesi gerekiyor.
- Sistem dokümantasyonu
  - Testlerin kendileri, kodun ne yapması gerektiğini açıklayan bir belge biçimidir.

Regresyon Testi



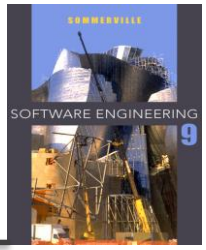
- Regresyon testi, değişikliklerin daha önce çalışan kodu 'bozmadığını' kontrol etmek için sistemi test ediyor.
- Manuel bir test sürecinde, regresyon testi pahalıdır, ancak otomatik test ile basit ve anlaşılırdır. Programda her değişiklik yapıldığında tüm testler yeniden çalıştırılır.
- Değişiklik gerçekleştirilmeden önce testler 'başarılı' çalıştırılmalıdır.

Sürüm Testi



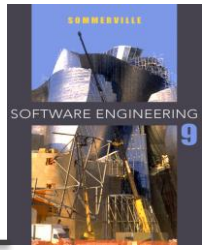
- Sürüm testi, geliştirme ekibinin dışında kullanılması amaçlanan bir sistemin belirli bir sürümünü test etme sürecidir.
- Sürüm testi sürecinin birincil amacı, sistemin tedarikçisini kullanım için yeterince iyi olduğuna ikna etmektir.
  - Bu nedenle sürüm testi, sistemin belirtilen işlevselliğini, performansını ve güvenilirliğini sağladığını ve normal kullanım sırasında başarısız olmadığını göstermelidir.
- Serbest bırakma testi, genellikle testlerin yalnızca sistem spesifikasyonundan türetildiği bir kara kutu test sürecidir.

Sürüm (Yayın) Testi Ve Sistem Testi



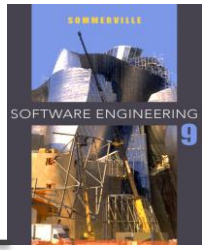
- Yayın testi, bir sistem testi biçimidir.
- Önemli farklılıklar:
  - Sistem geliştirmeye dahil olmayan ayrı bir ekip, sürüm testinden sorumlu olmalıdır.
  - Geliştirme ekibi tarafından yapılan sistem testi, sistemdeki hataları keşfetmeye odaklanmalıdır (hata testi). Yayın/sürüm testinin amacı, sistemin gereksinimlerini karşılayıp karşılamadığını ve harici kullanım için yeterince iyi olup olmadığını kontrol etmektir (doğrulama testi).

Gereksinimlere Dayalı Test



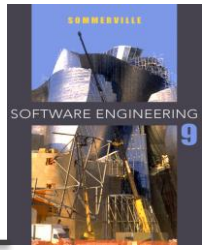
- Gereksinim tabanlı test, her bir gereksinimi incelemeyi ve bunun için bir test veya test geliştirmeyi içerir.
- AK-HYS gereksinimleri:
  - Bir hastanın belirli bir ilaca alerjisi olduğu biliniyorsa, o ilacın reçetesi sistem kullanıcısına bir uyarı mesajı verilmesiyle sonuçlanacaktır.
  - Reçete yazan kişi alerji uyarısını görmezden gelmeyi seçerse, bunun neden göz ardı edildiğine dair bir neden sunmalıdır.

Gereksinim Testleri



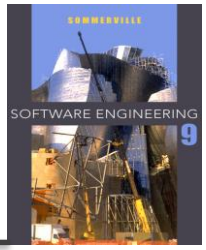
- Bilinen alerjisi olmayan bir hasta kaydı oluşturun. Var olduğu bilinen alerjiler için ilaç yazınız. Sistem tarafından bir uyarı mesajı verilmediğini kontrol edin.
- Bilinen bir alerjiyle hasta kaydı oluşturun. Hastanın alerjisi olduğu ilacı reçete edin ve uyarının sistem tarafından verilip verilmediğini kontrol edin.
- İki veya daha fazla ilaca karşı alerjilerin kaydedildiği bir hasta kaydı oluşturun. Bu ilaçların her ikisini de ayrı ayrı reçete edin ve her ilaç için doğru uyarının verildiğini kontrol edin.
- Hastanın alerjisi olan iki ilacı reçete edin. İki uyarının doğru şekilde verildiğini kontrol edin.
- Bir uyarı veren ve bu uyarıyı geçersiz kılan bir ilacı reçete edin. Sistemin, kullanıcının uyarının neden reddedildiğini açıklayan bilgi sağlamasını gerektirip gerektirmediğini kontrol edin.

Senaryoya Göre Test Edilen Özellikler



- Sistemde oturum açarak kimlik doğrulama.
- Belirtilen hasta kayıtlarının bir dizüstü bilgisayara indirilmesi ve yüklenmesi.
- Ev ziyareti planlaması.
- Bir mobil cihazda hasta kayıtlarının şifrelenmesi ve şifresinin çözülmesi.
- Kayıt alma ve değiştirme.
- Yan etki bilgilerini tutan ilaç veri tabanına bağlantılar.
- Çağrı yönlendirme sistemi.

AK-HYS İçin Bir Kullanım Senaryosu



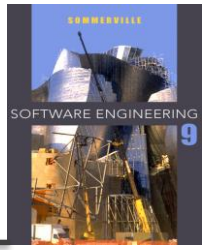
Kate, akıl sağlığı konusunda uzmanlaşmış bir hemşiredir. Sorumluluklarından biri, tedavilerinin etkili olup olmadığını ve ilaç yan etkilerinden muzdarip olmadıklarını kontrol etmek için hastaları evde ziyaret etmektir.

Kate, ev ziyaretleri için bir günde AK-HYS'de oturum açar ve ziyaret edilecek hastalarla ilgili özet bilgilerle birlikte o güne ait ev ziyaretleri programını yazdırmak için kullanır. Bu hastaların kayıtlarının dizüstü bilgisayarına indirilmesini talep ediyor. Dizüstü bilgisayardaki kayıtları şifrelemek için anahtar ifadesi istenir.

Ziyaret ettiği hastalardan biri, depresyon tedavisi gören Jim. Jim, ilacın kendisine yardımcı olduğunu hissediyor, ancak geceleri onu uyanık tutma yan etkisi olduğuna inanıyor. Kate, Jim'in kaydına bakar ve kaydın şifresini çözmek için anahtar ifadesini sorar. Reçete edilen ilacı kontrol eder ve yan etkilerini sorgular. Uykusuzluk bilinen bir yan etkidir, bu yüzden Jim'in kaydındaki sorunu not eder ve ilacını değiştirmek için kliniği ziyaret etmesini önerir. O, bir doktorla randevu almak için kliniğe döndüğünde Kate'in onu aramak için bir uyarıyı girmesini kabul eder. Görüşmeyi bitirir ve sistem Jim'in kaydını yeniden şifreler.

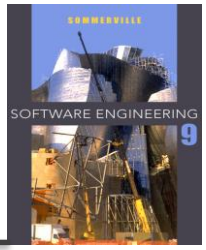
Kate, konsültasyonlarını bitirdikten sonra kliniğe geri döner ve ziyaret ettiği hastaların kayıtlarını veri tabanına yükler. Sistem, Kate için takip bilgileri için iletişime geçmesi ve klinik randevuları yapması gereken hastalar için bir çağrı listesi oluşturur.

Performans Testi



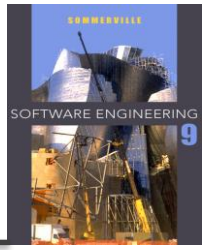
- Yayın testinin bir kısmı, performans ve güvenilirlik gibi bir sistemin ortaya çıkan özelliklerinin test edilmesini içerebilir.
- Testler, sistemin kullanım profilini yansıtmalıdır.
- Performans testleri genellikle, sistem performansı kabul edilemez hale gelene kadar yükün sürekli olarak arttırıldığı bir dizi testin planlanmasını içerir.
- Stres testi, sistemin hata davranışını test etmek için kasıtlı olarak aşırı yüklendiği bir tür performans testidir.

Kullanıcı Testi



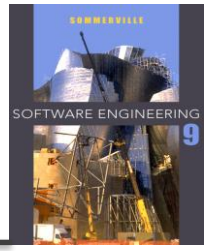
- Kullanıcı veya müşteri testi, kullanıcıların veya müşterilerin sistem testi hakkında girdi ve tavsiye sağladıkları test sürecinde bir aşamadır.
- Kapsamlı sistem ve sürüm testleri yapıldığında bile kullanıcı testi çok önemlidir.
  - Bunun nedeni, kullanıcının çalışma ortamından gelen etkilerin bir sistemin güvenilirliği, performansı, kullanılabilirliği ve sağlamlığı üzerinde büyük bir etkiye sahip olmasıdır. Bunlar bir test ortamında çoğaltılamaz.

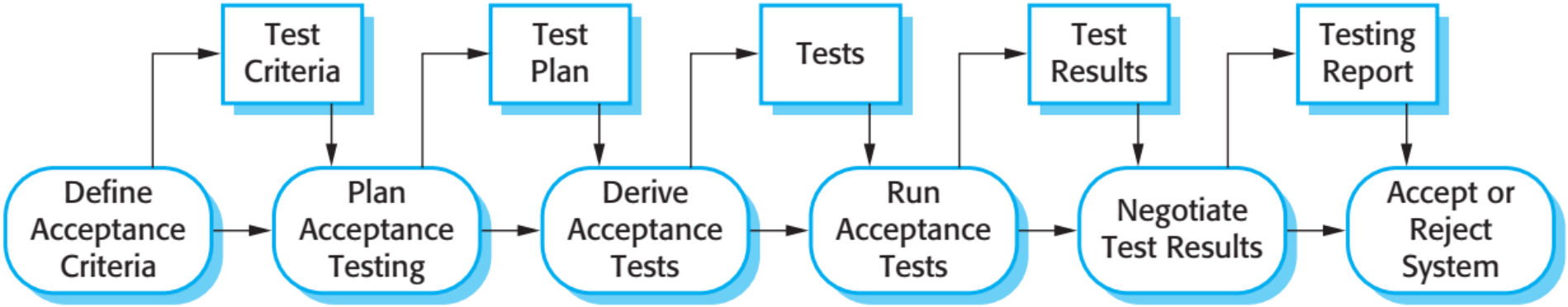
Kullanıcı Testi Türleri



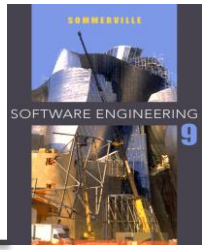
- Alfa testi
  - Yazılımın kullanıcıları, yazılımı geliştiricinin sitesinde test etmek için geliştirme ekibiyle birlikte çalışır.
- Beta testi
  - Kullanıcılara, deney yapmalarına ve sistem geliştiricileriyle keşfettikleri sorunları ortaya çıkarmalarına olanak sağlayan bir yazılım sürümü sunulur.
- Kabul testleri
  - Müşteriler, sistem geliştiricilerinden kabul edilmeye ve müşteri ortamında devreye alınmaya hazır olup olmadığına karar vermek için bir sistemi test eder. Öncelikle özel sistemler için.

Kabul Testi Süreci



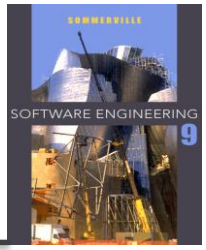


Kabul Testi Sürecindeki Aşamalar



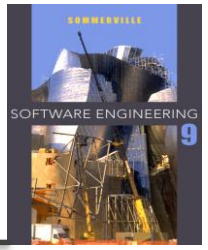
- Kabul kriterlerini tanımlayın
- Kabul testini planlayın
- Kabul testlerinin türetilmesi
- Kabul testlerini çalıştırın
- Test sonuçlarını görüşün
- Sistemi reddet / kabul et

Çevik Yöntemler ve Kabul Testi



- Çevik yöntemlerde, kullanıcı / müşteri geliştirme ekibinin bir parçasıdır ve sistemin kabul edilebilirliğine ilişkin kararlar vermekten sorumludur.
- Testler kullanıcı / müşteri tarafından tanımlanır ve değişiklik yapıldığında otomatik olarak çalıştırılmaları bakımından diğer testlerle entegre edilir.
- Ayrı bir kabul testi süreci yoktur.
- Buradaki temel sorun, yerleşik kullanıcının 'tipik' olup olmadığı ve tüm sistem paydaşlarının çıkarlarını temsil edip edemeyeceğidir.

Bölüm 2'nin Anahtar Noktaları



- Yazılımı test ederken, diğer sistemlerdeki kusurları keşfetmede etkili olan test senaryosu türlerini seçmek için deneyimi ve yönergeleri kullanarak yazılımı 'kırmaya/bozmaya' çalışmalısınız.
- Mümkün olan her yerde otomatik testler yazmalısınız. Testler, bir sistemde her değişiklik yapıldığında çalıştırılabilen bir programın içine yerleştirilmiştir.
- Önce test geliştirme, testlerin test edilecek koddan önce yazıldığı bir geliştirme yaklaşımıdır.
- Senaryo testi, tipik bir kullanım senaryosu icat etmeyi ve bunu test senaryolarını türetmek için kullanmayı içerir.
- Kabul testi, amacın yazılımın operasyonel ortamında dağıtılacak ve kullanılacak kadar iyi olup olmadığına karar vermek olduğu bir kullanıcı test sürecidir.