IT522 – Yazılım Mühendisliği 2021

PhD Furkan Gözükara, Toros University

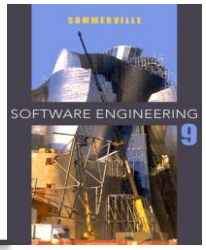
https://github.com/FurkanGozukara/Yazilim-Muhendisligi-IT522-2021

Ders 9

Yazılımın Gelişimi

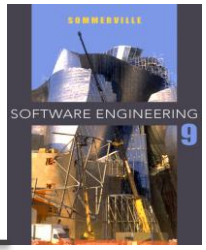
Kaynak : https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Presentations/index.html

Bölüm 1'de İşlenmiş Konular



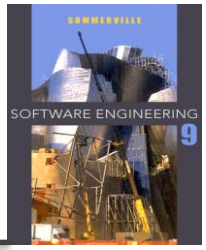
- Gelişim süreçleri
  - Yazılım sistemleri için süreçleri değiştirin
- Program gelişim dinamikleri
  - Yazılım gelişimini anlamak
- Yazılım bakımı
  - Operasyonel yazılım sistemlerinde değişiklik yapmak
- Eski sistem yönetimi
  - Yazılım değişikliği hakkında kararlar almak

Yazılımın Değişimi



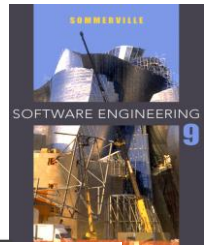
- Yazılım değişikliği kaçınılmazdır
  - Yazılım kullanıldığında yeni gereksinimler ortaya çıkar;
  - İş ortamı değişir;
  - Hataların onarılması gerekir;
  - Sisteme yeni bilgisayar ve ekipman eklenebilir;
  - Sistemin performansı veya güvenilirliğinin iyileştirilmesi gerekebilir.
- Tüm kuruluşlar için temel bir sorun, mevcut yazılım sistemlerinde değişiklik yapmak ve yönetmektir.

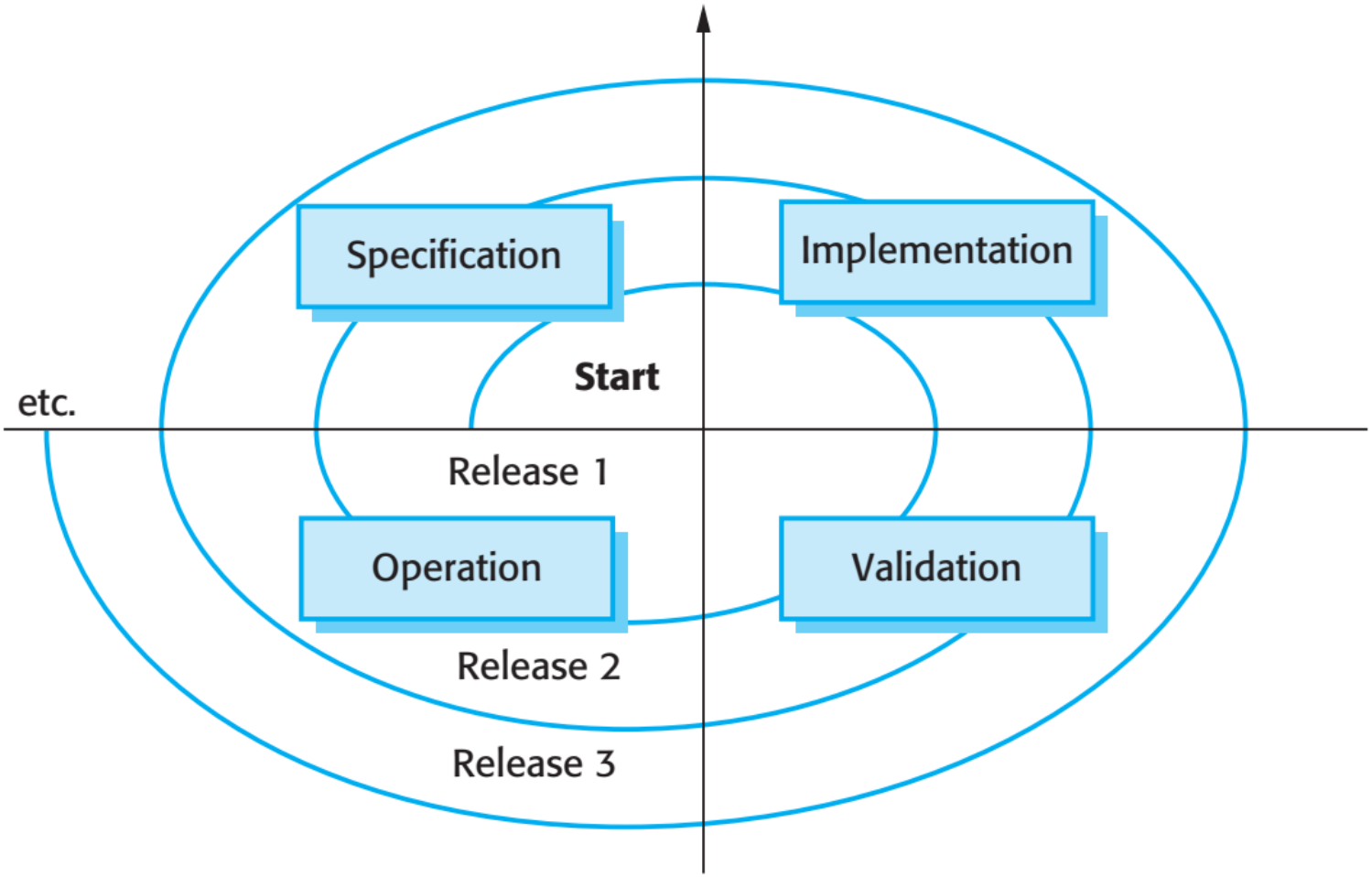
Gelişimin Önemi



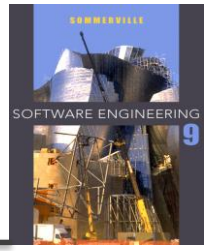
- Kuruluşların yazılım sistemlerine büyük yatırımları vardır - bunlar kritik ticari varlıklardır.
- Bu varlıkların işletme açısından değerini korumak için değiştirilmeleri ve güncellenmeleri gerekir.
- Büyük şirketlerde yazılım bütçesinin çoğu, yeni yazılım geliştirmek yerine mevcut yazılımları değiştirmeye ve geliştirmeye ayrılmıştır.

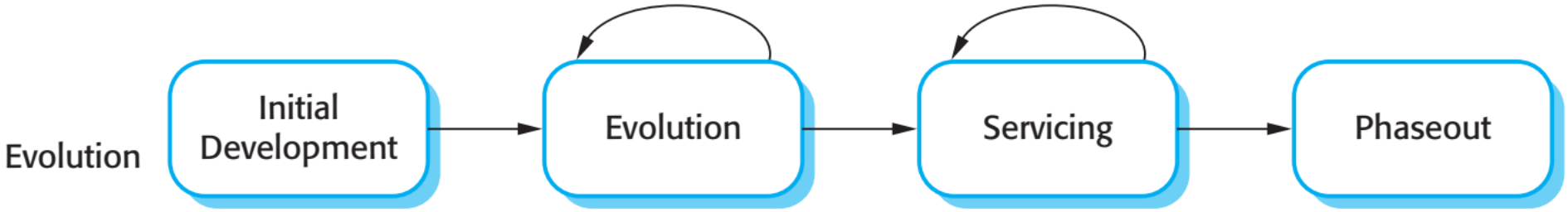
Sarmal Bir Gelişim ve Değişim Modeli



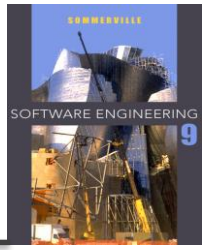


Gelişim ve Servis



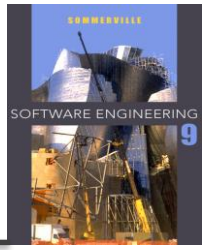


Gelişim ve Servis



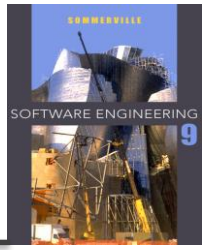
- Gelişim
  - Bir yazılım sisteminin yaşam döngüsünde operasyonel kullanımda olduğu ve sistemde yeni gereksinimler önerilip uygulandıkça evrildiği aşama.
- Servis
  - Bu aşamada, yazılım yararlı olmaya devam eder, ancak yapılan değişiklikler, onu çalışır durumda tutmak için gerekli olanlardır, yani hata düzeltmeleri ve yazılım ortamındaki değişiklikleri yansıtacak değişiklikler yapılır. Yeni işlev eklenmez.
- Aşamalı Kullanımdan Çıkma
  - Yazılım yine de kullanılabilir ancak üzerinde başka bir değişiklik yapılmaz.

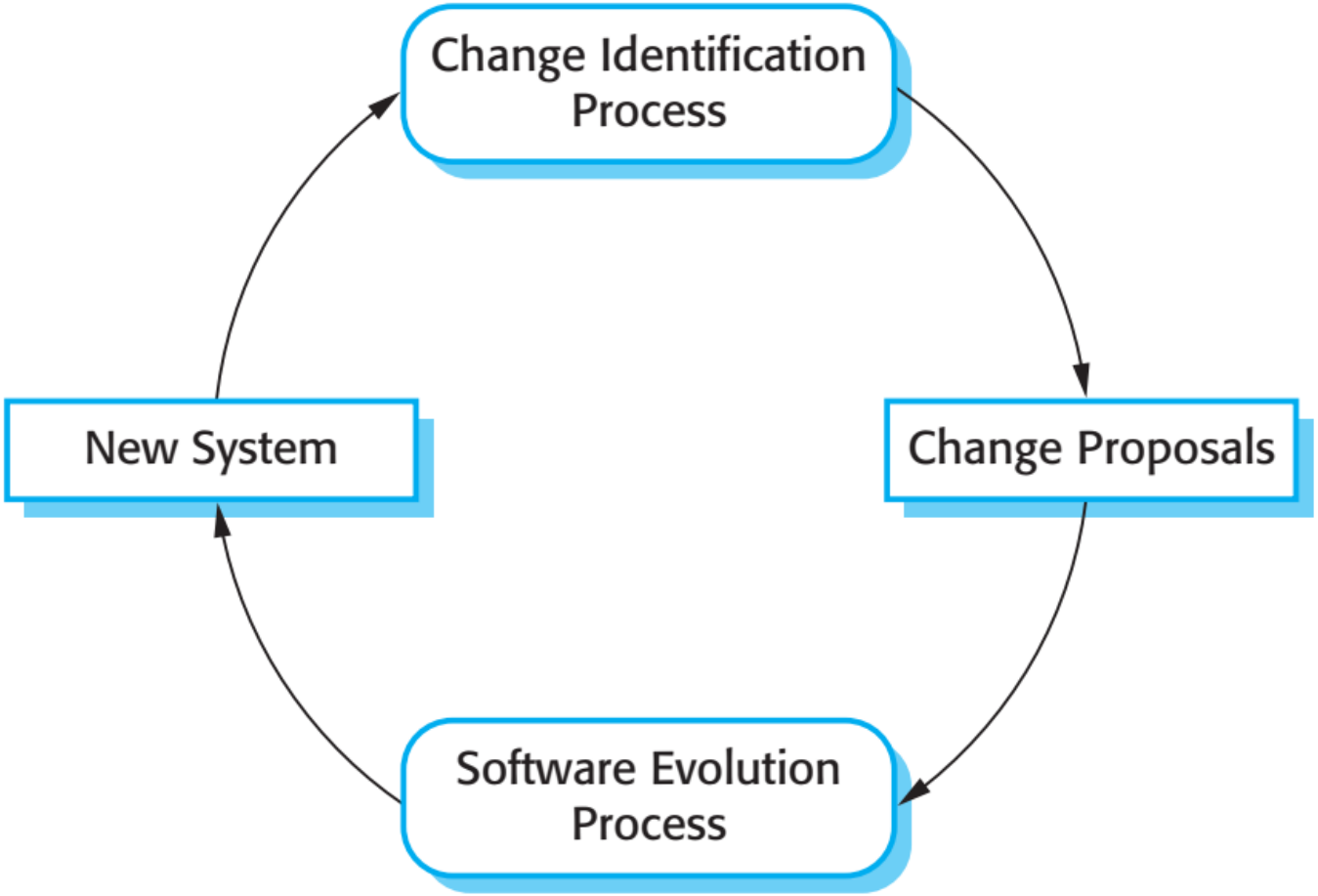
Gelişim Süreçleri



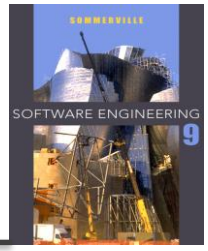
- Yazılım gelişim süreçleri şunlara bağlıdır:
  - Bakımı yapılan yazılımın türü;
  - Kullanılan geliştirme süreçleri;
  - İlgili kişilerin becerileri ve deneyimleri.
- Değişim önerileri, sistem gelişiminin itici gücüdür.
  - Değişiklikten etkilenen bileşenlerle bağlantılı olmalı, böylece değişikliğin maliyetinin ve etkisinin tahmin edilmesine izin verilmelidir.
- Değişiklik tanımlama ve gelişim, sistem ömrü boyunca devam eder.

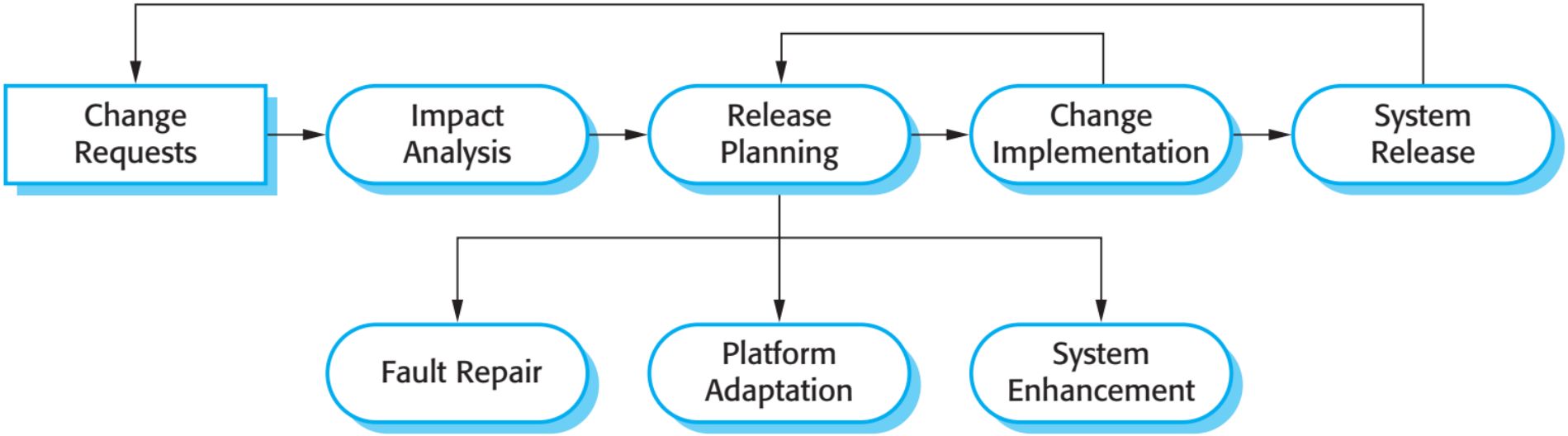
Tanımlama ve Gelişim Süreçlerini Değiştirin



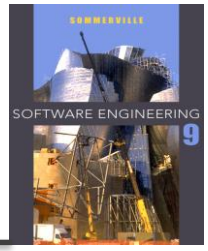


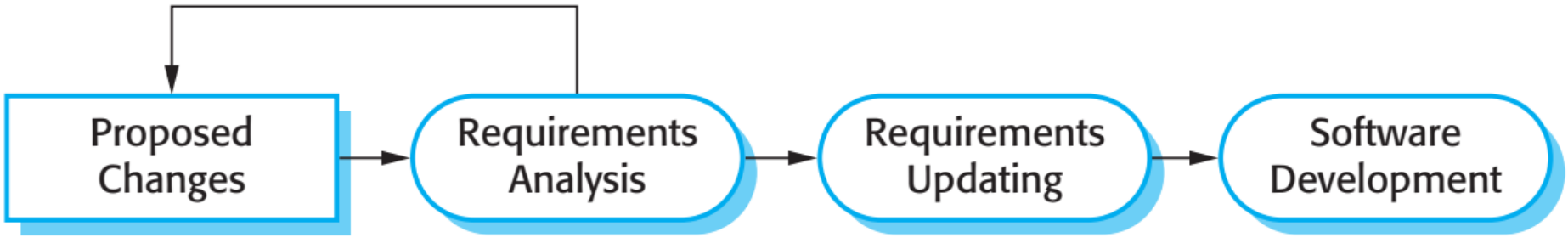
Yazılım Geliştirme Süreci



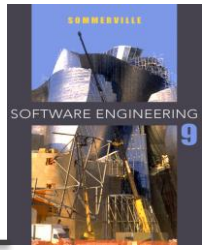


İmplementasyon Değiştir



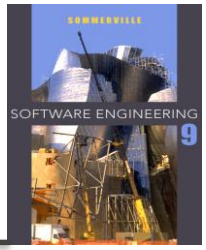


İmplementasyon Değiştir



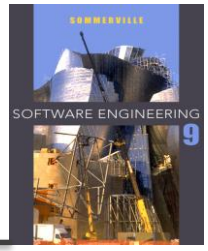
- Sistemdeki revizyonların tasarlandığı, uygulandığı ve test edildiği geliştirme sürecinin tekrarı.
- Kritik bir fark, değişiklik uygulamasının ilk aşamasının, özellikle de değişiklik uygulamasından orijinal sistem geliştiricileri sorumlu değilse, programın anlaşılmasını içerebilmesidir.
- Programı anlama aşamasında, programın nasıl yapılandırıldığını, nasıl işlevsellik sağladığını ve önerilen değişikliğin programı nasıl etkileyebileceğini anlamanız gerekir.

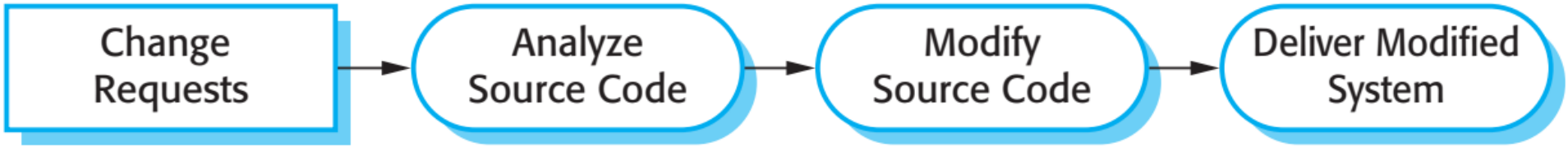
Acil Değişiklik Talepleri



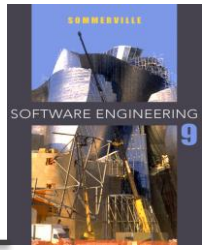
- Yazılım mühendisliği sürecinin tüm aşamalarından geçmeden acil değişikliklerin uygulanması gerekebilir
  - Normal çalışmanın devam etmesini sağlamak için ciddi bir sistem arızasının onarılması gerekiyorsa;
  - Sistem ortamındaki değişikliklerin (örneğin bir işletim sistemi yükseltmesi) beklenmeyen etkileri varsa;
  - Çok hızlı yanıt gerektiren iş değişiklikleri varsa (örneğin, rakip bir ürünün piyasaya sürülmesi).

Acil Onarım Süreci



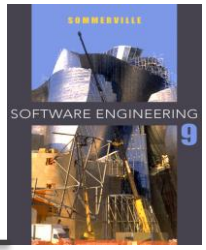


Çevik Yöntemler ve Gelişim



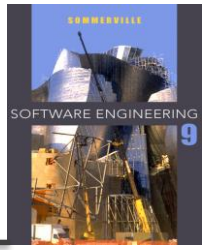
- Çevik yöntemler artımlı gelişime dayalıdır, bu nedenle geliştirmeden gelişime geçiş sorunsuzdur.
  - Gelişim, basitçe, sık sistem sürümlerine dayanan geliştirme sürecinin bir devamıdır.
- Otomatik regresyon testi, bir sistemde değişiklik yapıldığında özellikle değerlidir.
- Değişiklikler ek kullanıcı hikayeleri olarak ifade edilebilir.

Devir Sorunları



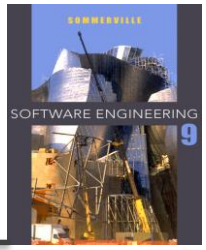
- Geliştirme ekibinin çevik bir yaklaşım kullandığı ancak gelişim ekibinin çevik yöntemlere aşina olmadığı ve plan tabanlı bir yaklaşımı tercih ettiği durumlarda.
  - Gelişim ekibi, gelişimi desteklemek için ayrıntılı dokümantasyon bekleyebilir ve bu, çevik süreçlerde üretilmez.
- Geliştirme için plana dayalı bir yaklaşımın kullanıldığı, ancak gelişim ekibinin çevik yöntemlerini kullanmayı tercih ettiği durumlarda.
  - Gelişim ekibinin otomatik testler geliştirmeye sıfırdan başlaması gerekebilir ve sistemdeki kod, çevik geliştirmede beklendiği gibi yeniden düzenlenmemiş ve basitleştirilmemiş olabilir.

Program Gelişim Dinamikleri



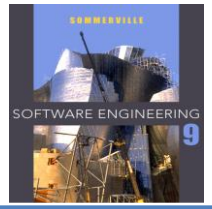
- *Program gelişim dinamikleri*, sistem değişikliği süreçlerinin incelenmesidir.
- Birkaç büyük ampirik çalışmadan sonra, Lehman ve Belady, gelişim geçirirken tüm sistemlere uygulanan bir dizi 'yasa' olduğunu öne sürdüler.
- Kanunlar yerine mantıklı gözlemler var. Büyük kuruluşlar tarafından geliştirilen büyük sistemlere uygulanabilir.
  - Bunların diğer yazılım sistemi türleri için geçerli olup olmadığı açık değildir.

Değişim Kaçınılmaz



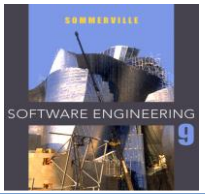
- Ortam değiştiği için sistem geliştirilirken sistem gereksinimlerinin değişmesi muhtemeldir. Bu nedenle, teslim edilen bir sistem gereksinimlerini karşılamayacaktır!
- Sistemler çevreleriyle sıkı sıkıya bağlıdır. Bir sistem bir ortama kurulduğunda, o ortamı ve dolayısıyla sistem gereksinimlerini değiştirir.
- Bir ortamda yararlı kalmaları için sistemler DEĞİŞTİRİLMELİDİR.

Lehman Yasaları



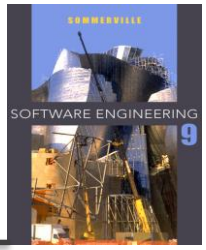
Yasa	Açıklama
Devam eden değişim	Gerçek dünya ortamında kullanılan bir program mutlaka değişmeli veya bu ortamda giderek daha az kullanışlı hale gelecektir.
Artan karmaşıklık	Gelişen bir program değiştikçe, yapısı daha karmaşık hale gelme eğilimindedir. Yapının korunmasına ve basitleştirilmesine ekstra kaynaklar ayrılmalıdır.
Büyük program gelişimi	Program gelişimi, kendi kendini düzenleyen bir süreçtir. Boyut, sürümler arasındaki süre ve bildirilen hataların sayısı gibi sistem öznitelikleri, her sistem sürümü için yaklaşık olarak değişmez.
Örgütsel istikrar	Bir programın ömrü boyunca, geliştirme hızı yaklaşık olarak sabittir ve sistem geliştirmeye ayrılan kaynaklardan bağımsızdır.

Lehman Yasaları



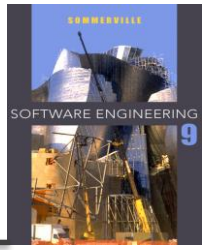
Yasa	Açıklama
Aşinalığın korunması	Bir sistemin ömrü boyunca, her sürümdeki artımlı değişiklik yaklaşık olarak sabittir.
Devam eden büyüme	Sistemlerin sunduğu işlevsellik, kullanıcı memnuniyetini korumak için sürekli olarak artmalıdır.
Düşen kalite	İşletim ortamlarındaki değişiklikleri yansıtacak şekilde değiştirilmedikleri sürece sistemlerin kalitesi düşecektir.
Geri bildirim sistemi	Gelişim süreçleri, çok ajanlı, çok döngülü geri bildirim sistemlerini içerir ve önemli ürün iyileştirmesi elde etmek için bunları geri bildirim sistemleri olarak ele almanız gerekir.

Lehman Yasalarının Uygulanabilirliği

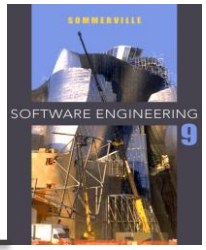


- Lehman yasaları, büyük kuruluşlar tarafından geliştirilen büyük, özel sistemlere genel olarak uygulanabilir görünmektedir.
  - 2000'li yılların başında Lehman'ın FEAST projesi üzerinde çalışmasıyla onaylandı.
- Nasıl değiştirilmeleri gerektiği açık değil
  - Küçültülmüş yazılım ürünleri;
  - Önemli sayıda COTS bileşeni içeren sistemler;
  - Küçük kuruluşlar;
  - Orta ölçekli sistemler.

Bölüm 1'in Anahtar Noktaları



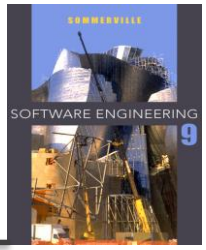
- Yazılım geliştirme ve gelişim, spiral bir model kullanılarak temsil edilebilen entegre, yinelemeli bir süreç olarak düşünülebilir.
- Özel sistemler için, yazılım bakım maliyetleri genellikle yazılım geliştirme maliyetlerini aşar.
- Yazılım geliştirme süreci, değişiklik talepleri tarafından yönlendirilir ve değişiklik etki analizi, sürüm planlaması ve değişiklik uygulamasını içerir.
- Lehman yasaları, değişimin sürekli olduğu nosyonu gibi, sistem gelişimi üzerine uzun dönemli çalışmalardan elde edilen bir dizi kavrayışı tanımlar.



Ders 9 - Yazılımın Gelişimi

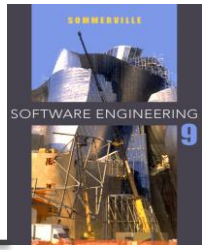
Bölüm 2

Yazılım Bakımı



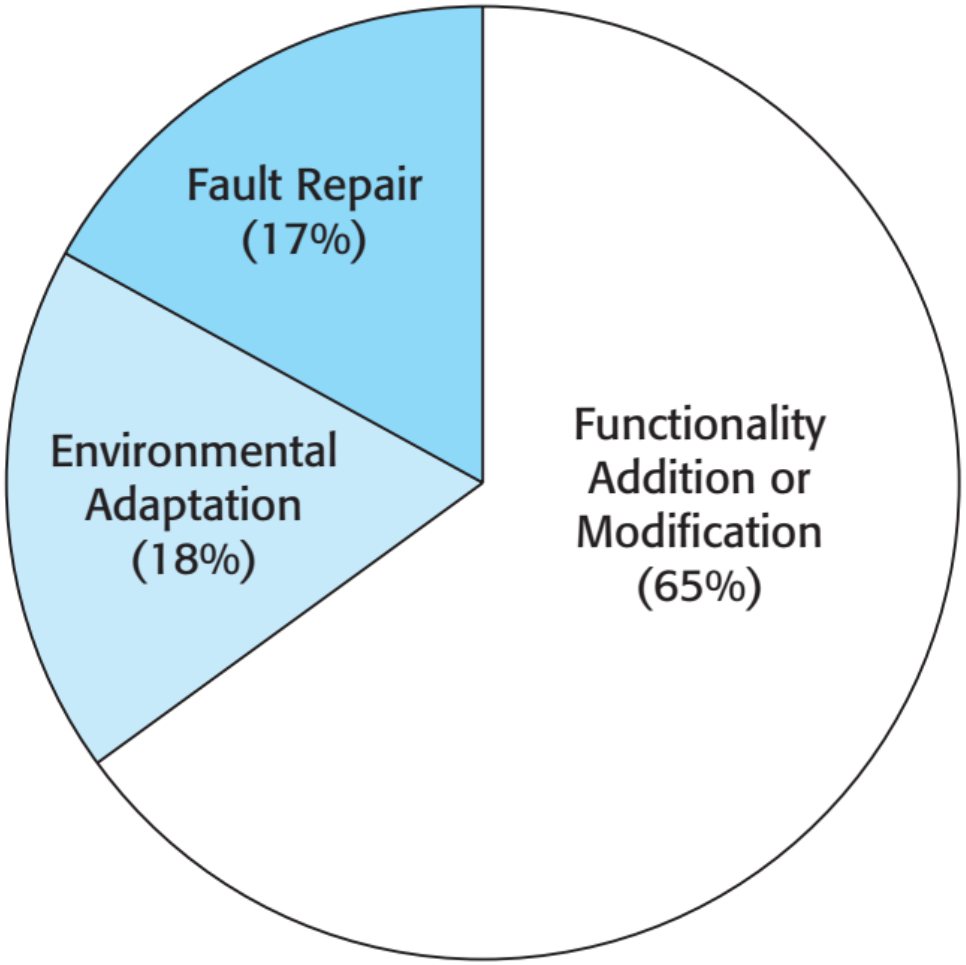
- Kullanıldıktan sonra bir programı değiştirme.
- Terim çoğunlukla özel yazılımı değiştirmek için kullanılır. Genel yazılım ürünlerinin yeni sürümler oluşturmak için geliştiği söyleniyor.
- Bakım normalde sistemin mimarisinde büyük değişiklikler içermez.
- Değişiklikler, mevcut bileşenleri değiştirerek ve sisteme yeni bileşenler ekleyerek gerçekleştirilir.

Bakım Türleri

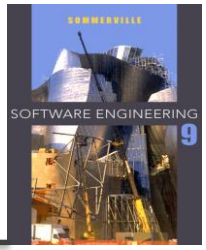


- Yazılım hatalarını onarmak için bakım
  - Bir sistemi, karşılaşılan eksiklikleri düzeltecek şekilde değiştirmek ve yazılımın gereksinimlerini karşılamak.
- Yazılımı farklı bir işletim ortamına uyarlamak için bakım
  - Bir sistemi, ilk uygulamasından farklı bir ortamda (bilgisayar, işletim sistemi vb.) çalışacak şekilde değiştirme.
- Sistemin işlevsellik eklemek veya işlevselliğini değiştirmek için bakım
  - Yeni gereksinimleri karşılamak için sistemi değiştirmek.

Şekil 9.8 Bakım Efor Dağılımı

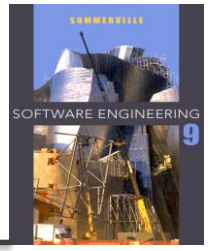


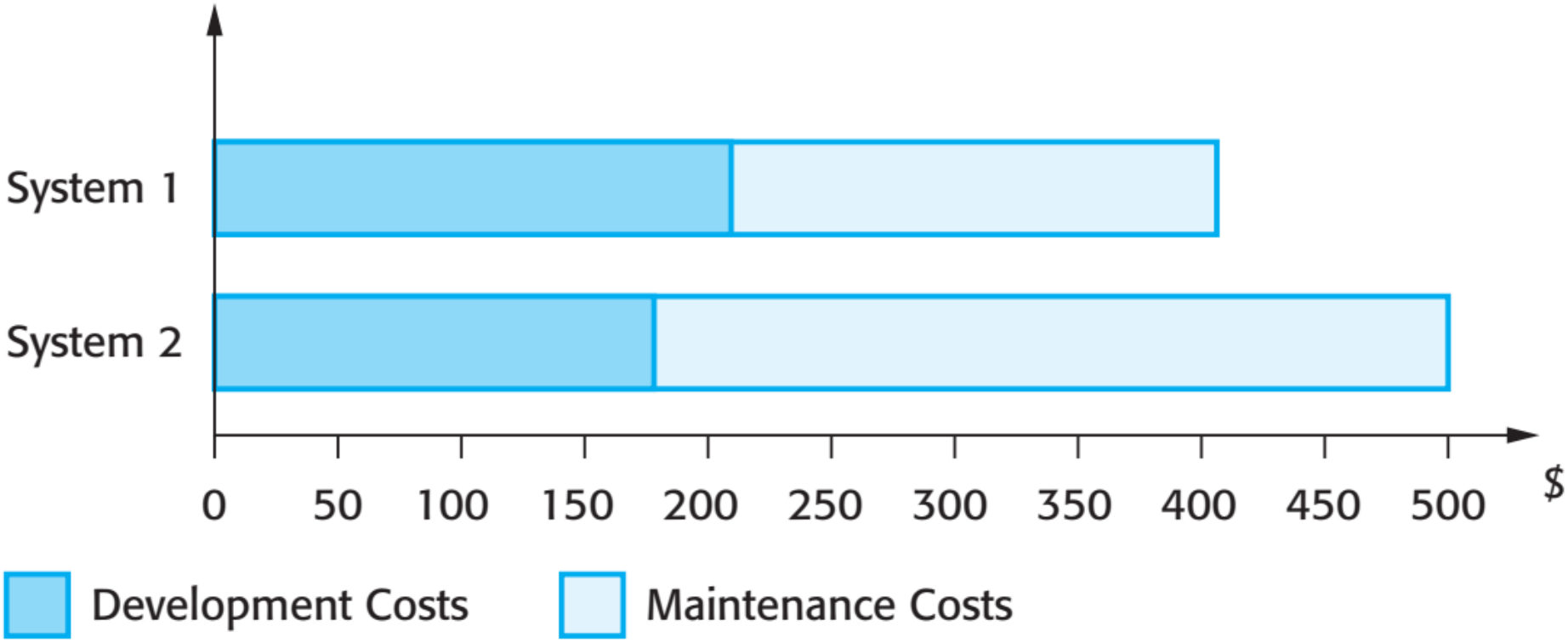
Bakım Maliyetleri



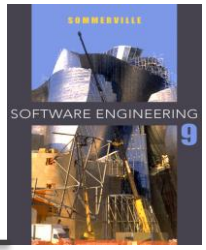
- Genellikle geliştirme maliyetlerinden daha yüksektir (uygulamaya bağlı olarak 2 * ila 100 *).
- Hem teknik hem de teknik olmayan faktörlerden etkilenir.
- Yazılım sürdürüldükçe artar. Bakım, yazılım yapısını bozduğundan daha fazla bakımı zorlaştırır.
- Yaşlanan yazılımların yüksek destek maliyetleri olabilir (örneğin eski diller, derleyiciler vb.).

Şekil 9.9 Geliştirme ve Bakım Maliyetleri



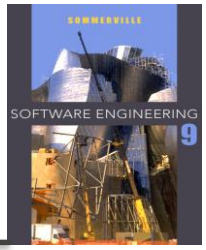


Bakım Maliyeti Faktörleri



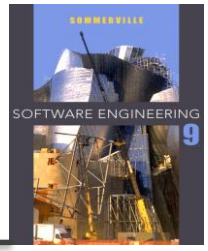
- Takım istikrarı
  - Bir süre aynı personel ile birlikte bakım yapılırsa bakım maliyetleri azalır.
- Sözleşmeden doğan sorumluluk
  - Bir sistemin geliştiricilerinin bakım için sözleşmeye dayalı bir sorumluluğu olmayabilir, bu nedenle gelecekteki değişiklikler için tasarlama konusunda herhangi bir teşvik yoktur.
- Personel becerileri
  - Bakım personeli genellikle deneyimsizdir ve sınırlı alan bilgisine sahiptir.
- Program yaşı ve yapısı
  - Programlar yaşlandıkça yapıları bozulur ve anlaşılması ve değiştirilmesi zorlaşır.

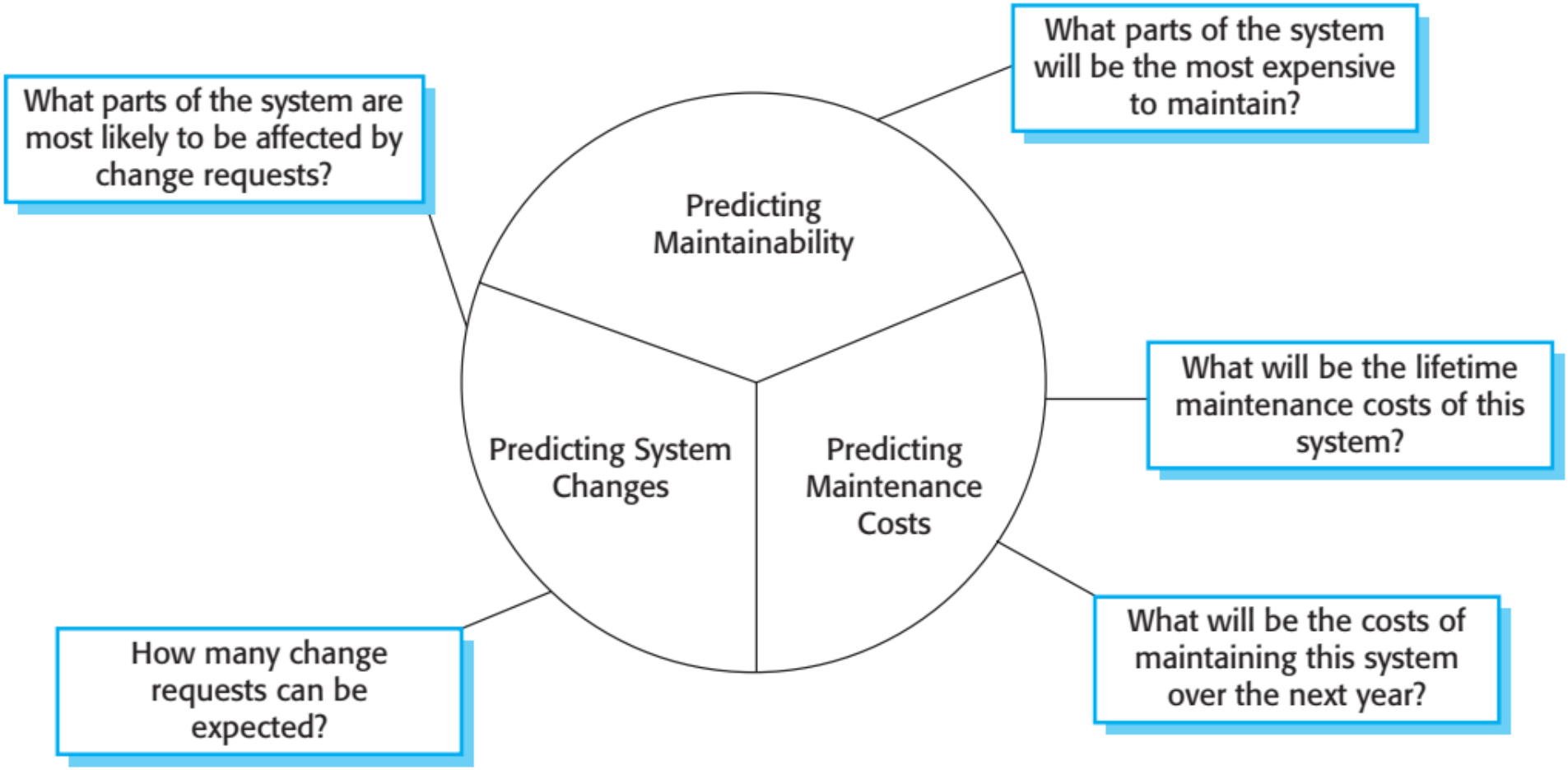
Bakım Tahmini



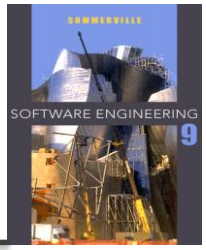
- Bakım tahmini, sistemin hangi parçalarının sorunlara neden olabileceğinin ve bakım maliyetlerinin yüksek olabileceğinin değerlendirilmesiyle ilgilidir.
  - Değişikliğin kabulü, değişiklikten etkilenen bileşenlerin sürdürülebilirliğine bağlıdır;
  - Değişikliklerin uygulanması sistemin kalitesini düşürür ve sürdürülebilirliğini azaltır;
  - Bakım maliyetleri, değişikliklerin sayısına bağlıdır ve değişim maliyetleri, bakım yapılabilirliğe bağlıdır.

Bakım Tahmini



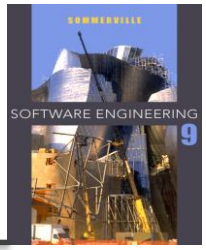


Tahmini Değiştir



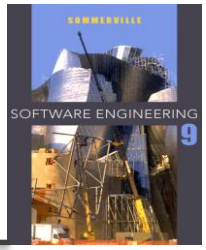
- Değişikliklerin sayısını tahmin etmek, bir sistem ile çevresi arasındaki ilişkilerin anlaşılmasını gerektirir.
- Sıkıca bağlı sistemler, ortam her değiştiğinde değişiklik gerektirir.
- Bu ilişkiyi etkileyen faktörler şunlardır:
  - Sistem arayüzlerinin sayısı ve karmaşıklığı;
  - Doğası gereği değişken olan sistem gereksinimlerinin sayısı;
  - Sistemin kullanıldığı iş süreçleri.

Karmaşıklık Ölçümleri



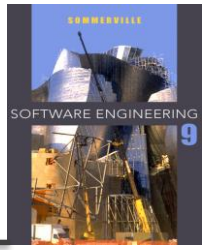
- Sürdürülebilirlik tahminleri, sistem bileşenlerinin karmaşıklığı değerlendirilerek yapılabilir.
- Çalışmalar, çoğu bakım çabasının nispeten az sayıda sistem bileşeni üzerinde harcandığını göstermiştir.
- Karmaşıklık şunlara bağlıdır:
  - Kontrol yapılarının karmaşıklığı;
  - Veri yapılarının karmaşıklığı;
  - Nesne, yöntem (prosedür) ve modül boyutu.

Süreç Ölçütleri



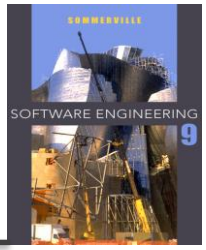
- Sürdürülebilirliği değerlendirmek için süreç ölçütleri kullanılabilir
  - Düzeltici bakım taleplerinin sayısı;
  - Etki analizi için gereken ortalama süre;
  - Bir değişiklik talebini uygulamak için geçen ortalama süre;
  - Bekleyen değişiklik taleplerinin sayısı.
- Bunlardan herhangi biri veya tümü artıyorsa, bu, sürdürülebilirlikte bir düşüşe işaret edebilir.

Sistem Yeniden Mühendisliği



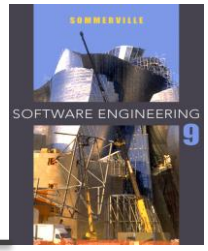
- Eski bir sistemin bir bölümünü veya tamamını işlevselliğini değiştirmeden yeniden yapılandırma veya yeniden yazma.
- Daha büyük bir sistemin tüm alt sistemleri olmasa da bazılarının sık bakım gerektirdiği durumlarda uygulanabilir.
- Yeniden mühendislik, bakımını kolaylaştırmak için çaba eklemeyi içerir. Sistem yeniden yapılandırılabilir ve yeniden belgelendirilebilir.

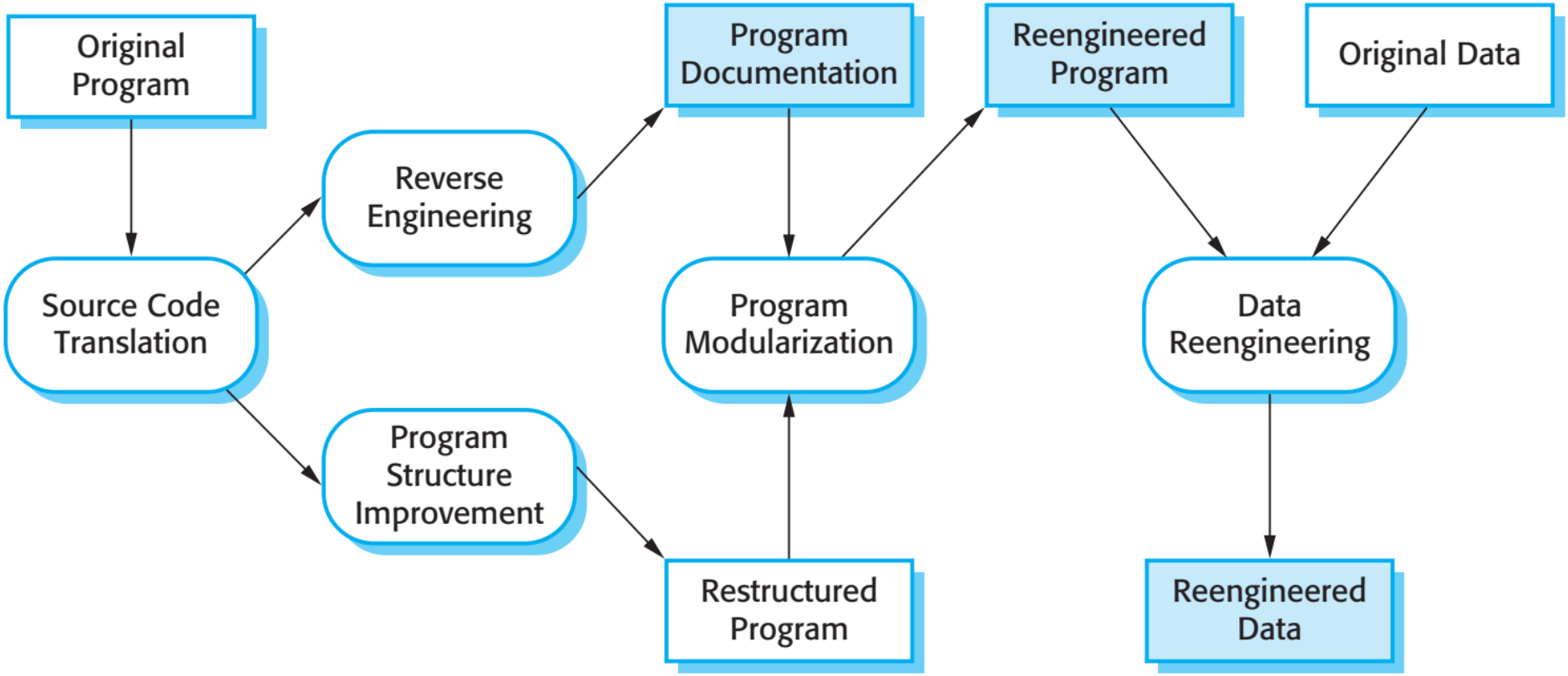
Yeniden Yapılandırmanın Avantajları



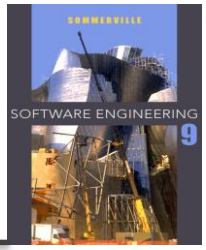
- Azaltılmış risk
  - Yeni yazılım geliştirmede yüksek risk vardır. Geliştirme sorunları, personel sorunları ve özellik sorunları olabilir.
- Düşük maliyet
  - Yeniden mühendisliğin maliyeti genellikle yeni yazılım geliştirme maliyetlerinden önemli ölçüde daha düşüktür.

Yeniden Yapılandırma Süreci



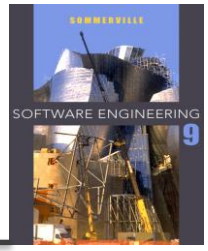


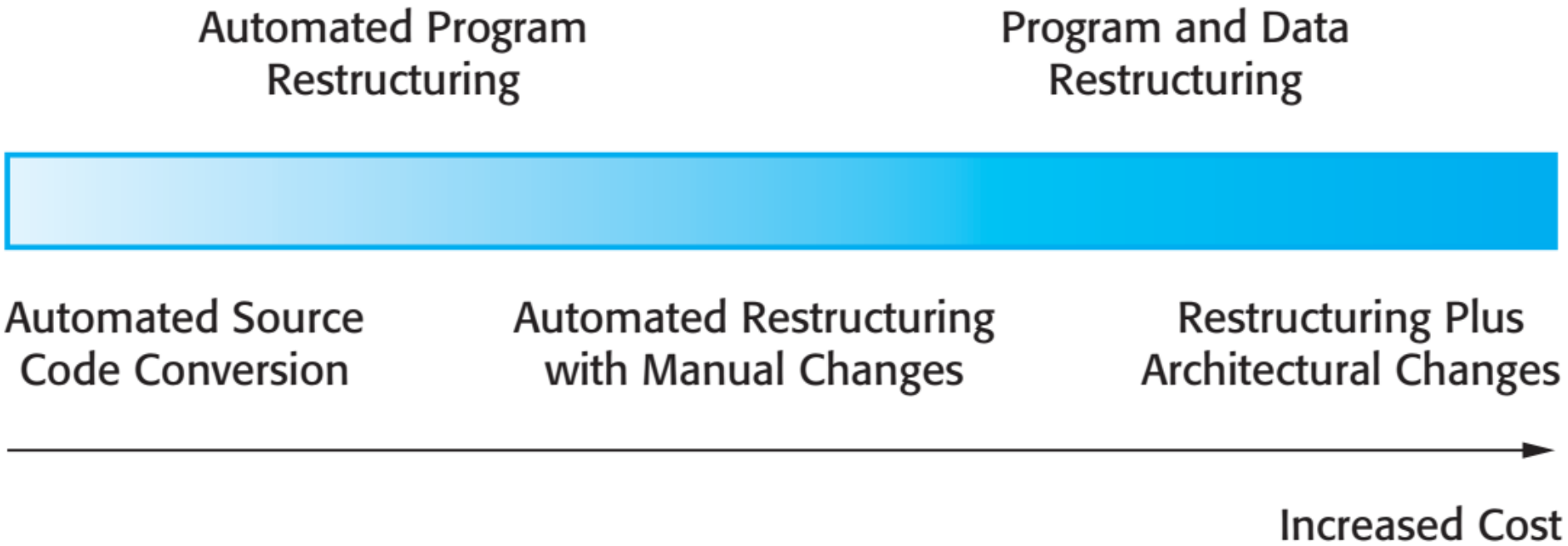
Yeniden Yapılandırma Süreci Faaliyetleri



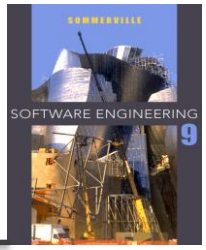
- Kaynak kod çevirisi
  - Kodu yeni bir dile dönüştürün.
- Tersine mühendislik
  - Programı anlamak için analiz edin;
- Program yapısı iyileştirme
  - Anlaşılabilirlik için otomatik olarak yeniden yapılandırın;
- Program modülerleştirme
  - Program yapısını yeniden düzenleyin;
- Veri yeniden yapılandırması
  - Sistem verilerini temizleyin ve yeniden yapılandırın.

Şekil 9.12 Yeniden Yapılandırma Yaklaşımları



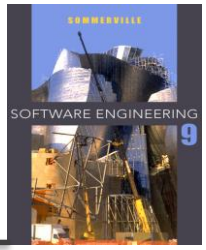


Maliyet Faktörlerinin Yeniden Yapılandırılması



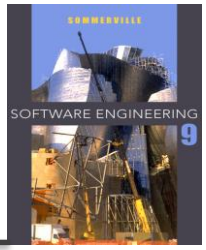
- Yeniden yapılandırılacak yazılımın kalitesi.
- Yeniden yapılandırma için mevcut alet desteği.
- Gerekli olan veri dönüşümünün kapsamı.
- Yeniden yapılandırma için uzman personelin mevcudiyeti.
  - Bu, artık yaygın olarak kullanılmayan teknolojiye dayalı eski sistemlerde bir sorun olabilir.

Yeniden Düzenleme Yoluyla Önleyici Bakım



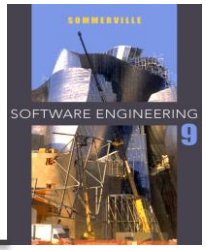
- Yeniden düzenleme, değişim yoluyla bozulmayı yavaşlatmak için bir programda iyileştirmeler yapma sürecidir.
- Yeniden düzenlemeyi, gelecekteki değişimin sorunlarını azaltan 'önleyici bakım' olarak düşünebilirsiniz.
- Yeniden düzenleme, yapısını iyileştirmek, karmaşıklığını azaltmak veya anlaşılmasını kolaylaştırmak için bir programı değiştirmeyi içerir.
- Bir programı yeniden düzenlediğinizde, işlevsellik eklememeli, bunun yerine program geliştirmeye odaklanmalısınız.

Yeniden Düzenleme ve Yeniden Yapılandırma



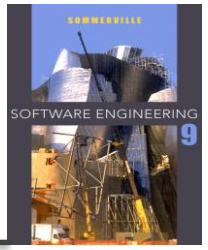
- Yeniden mühendislik, bir sisteme bir süre bakım yapıldıktan ve bakım maliyetleri arttıktan sonra gerçekleşir. Bakımı daha kolay yeni bir sistem oluşturmak için eski bir sistemi işlemek ve yeniden yapılandırmak için otomatik araçlar kullanırsınız.
- Yeniden düzenleme, geliştirme ve geliştirme süreci boyunca sürekli bir iyileştirme sürecidir. Bir sistemi sürdürmenin maliyetlerini ve zorluklarını artıran yapı ve kod bozulmasının önüne geçilmesi amaçlanmıştır.

Program Kodunda 'Kötü Kokular'



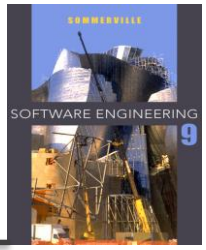
- Yinelenen kod
  - Aynı veya çok benzer kod, bir programın farklı yerlerinde bulunabilir. Bu, gerektiği gibi çağrılan tek bir yöntem veya işlev olarak kaldırılabilir ve uygulanabilir.
- Uzun yöntemler
  - Bir yöntem çok uzunsa, bir dizi daha kısa yöntem olarak yeniden tasarlanmalıdır.
- Anahtar (büyük / küçük harf) ifadeleri
  - Bunlar genellikle, anahtarın bir değerin türüne bağlı olduğu yinelemeyi içerir. Switch ifadeleri bir programın etrafına dağılmış olabilir. Nesne yönelimli dillerde, aynı şeyi elde etmek için genellikle çok biçimlilik kullanabilirsiniz.

Program Kodunda 'Kötü Kokular'



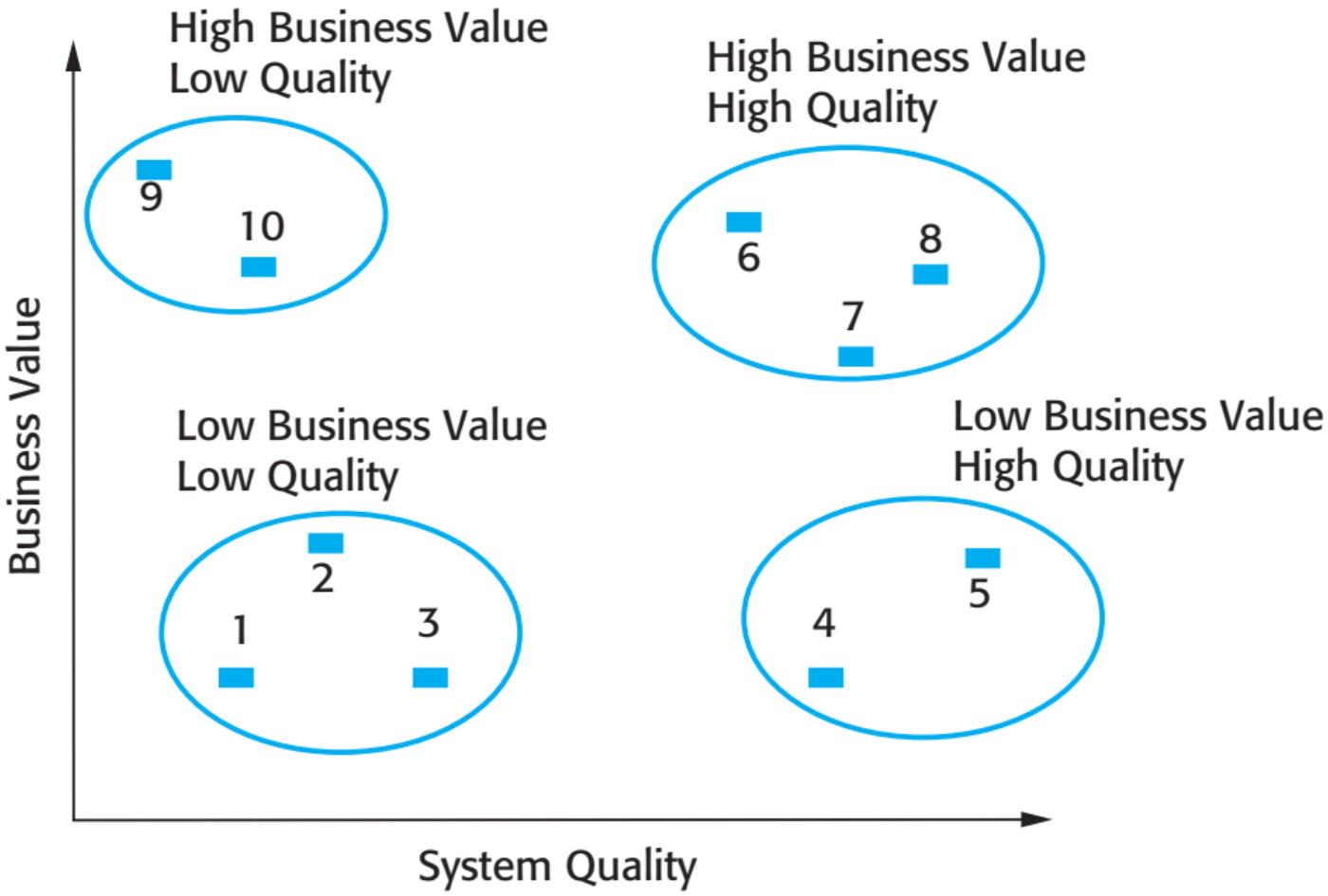
- Veri kümeleme
  - Veri kümeleri, aynı veri öğeleri grubu (sınıflardaki alanlar, yöntemlerdeki parametreler) bir programın birkaç yerinde yeniden meydana geldiğinde meydana gelir. Bunlar genellikle tüm verileri kapsayan bir nesne ile değiştirilebilir.
- Spekülatif genellik
  - Bu, geliştiriciler gelecekte gerekli olması durumunda bir programa genelliği dahil ettiğinde ortaya çıkar. Bu genellikle basitçe kaldırılabilir.

Eski Sistem Yönetimi

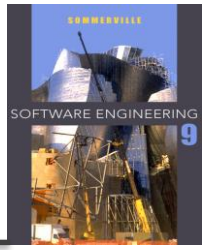


- Eski sistemlere güvenen kuruluşlar, bu sistemleri geliştirmek için bir strateji seçmelidir
  - Sistemi tamamen parçalayın ve iş süreçlerini artık gerekli olmayacak şekilde değiştirin;
  - Sistemin bakımını yapmaya devam edin;
  - Sürdürülebilirliğini artırmak için sistemi yeniden mühendislik yoluyla dönüştürün;
  - Sistemi yeni bir sistemle değiştirin.
- Seçilen strateji, sistem kalitesine ve iş değerine bağlı olmalıdır.

Şekil 9.13 Eski Sistem Değerlendirmesine Bir Örnek

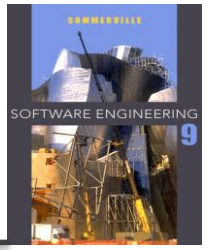


Eski Sistem Kategorileri



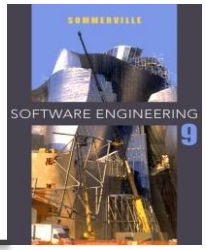
- Düşük kalite, düşük işletme değeri
  - Bu sistemler hurdaya çıkarılmalıdır.
- Düşük kaliteli, yüksek işletme değeri
  - Bunlar önemli bir iş katkısı sağlar ancak bakımı pahalıdır. Uygun bir sistem mevcutsa yeniden tasarlanmalı veya değiştirilmelidir.
- Yüksek kaliteli, düşük işletme değeri
  - COTS ile değiştirin, tamamen hurdaya ayırın veya bakımını yapın.
- Yüksek kaliteli, yüksek iş değeri
  - Normal sistem bakımını kullanarak çalışmaya devam edin.

İş Değeri Değerlendirmesi



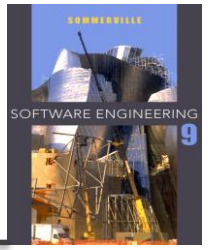
- Değerlendirme farklı bakış açılarını hesaba katmalıdır
  - Sistem son kullanıcıları;
  - Ticari Müşteriler;
  - Çizgi yöneticileri;
  - BT yöneticileri;
  - Üst düzey yöneticiler.
- Farklı paydaşlarla görüşün ve sonuçları harmanlayın.

İşletme Değeri Değerlendirmesindeki Sorunlar



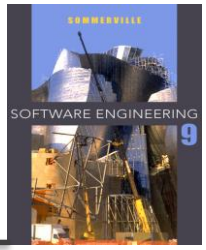
- Sistemin kullanımı
  - Sistemler yalnızca ara sıra veya az sayıda kişi tarafından kullanılıyorsa, düşük bir ticari değere sahip olabilirler.
- Desteklenen iş süreçleri
  - Bir sistem, verimsiz iş süreçlerinin kullanılmasını zorlarsa düşük bir iş değerine sahip olabilir.
- Sistem güvenilirliği
  - Bir sistem güvenilir değilse ve sorunlar ticari müşterileri doğrudan etkiliyorsa, sistemin iş değeri düşüktür.
- Sistem çıktıları
  - İş, sistem çıktılarına bağlıysa, sistemin yüksek bir iş değeri vardır.

Sistem Kalite Değerlendirmesi



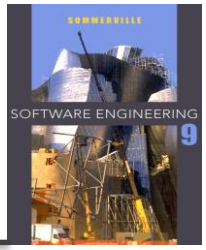
- İş süreci değerlendirmesi
  - İş süreci, işletmenin mevcut hedeflerini ne kadar iyi destekliyor?
- Çevre değerlendirmesi
  - Sistemin ortamı ne kadar etkili ve bakımı ne kadar pahalı?
- Uygulama değerlendirmesi
  - Uygulama yazılım sisteminin kalitesi nedir?

İş Süreci Değerlendirmesi



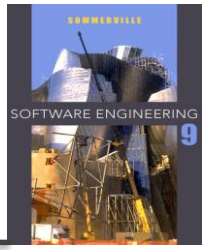
- Bakış açısına dayalı bir yaklaşım kullanın ve sistem paydaşlarından yanıtlar alın
  - Tanımlanmış bir süreç modeli var mı ve takip ediliyor mu?
  - Kuruluşun farklı bölümleri aynı işlev için farklı süreçler kullanıyor mu?
  - Süreç nasıl uyarlandı?
  - Diğer iş süreçleriyle ilişkiler nelerdir ve bunlar gerekli midir?
  - Süreç, eski uygulama yazılımı tarafından etkin bir şekilde destekleniyor mu?
- Örnek - bir seyahat sipariş sistemi, web tabanlı siparişin yaygın kullanımı nedeniyle düşük bir iş değerine sahip olabilir.

Çevre Değerlendirmesinde Kullanılan Faktörler



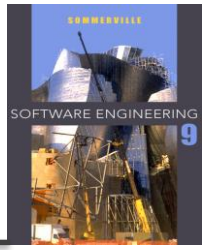
Faktör	Sorular
Tedarikçi istikrarı	Tedarikçi hala var mı? Tedarikçi finansal olarak istikrarlı mı ve varlığını sürdürme ihtimali var mı? Tedarikçi artık iş yapmıyorsa, sistemlerin bakımını başkası yapıyor mu?
Başarısızlık oranı	Donanımda yüksek oranda rapor edilmiş arıza var mı? Destek yazılımı çöküyor ve sistemi yeniden başlatmaya zorluyor mu?
Yaş	Donanım ve yazılım kaç yaşında? Donanım ve destek yazılımı ne kadar eski olursa, o kadar eski olacaktır. Hala düzgün çalışabilir ancak daha modern bir sisteme geçmenin önemli ekonomik ve ticari faydaları olabilir.
Verim	Sistemin performansı yeterli mi? Performans sorunlarının sistem kullanıcıları üzerinde önemli bir etkisi var mı?

Çevre Değerlendirmesinde Kullanılan Faktörler



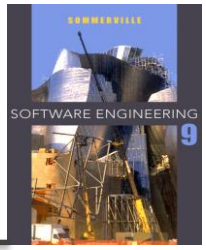
Faktör	Sorular
Destek gereksinimleri	Donanım ve yazılım için hangi yerel destek gerekiyor? Bu destekle ilgili yüksek maliyetler varsa, sistem değiştirmeyi düşünmeye değer olabilir.
Bakım maliyetleri	Donanım bakım ve destek yazılım lisanslarının maliyetleri nelerdir? Eski donanımlar, modern sistemlere göre daha yüksek bakım maliyetlerine sahip olabilir. Destek yazılımının yıllık lisans maliyetleri yüksek olabilir.
Birlikte çalışabilirlik	Sistemin diğer sistemlere bağlanmasında sorunlar var mı? Derleyiciler, örneğin, işletim sisteminin güncel sürümleriyle kullanılabilir mi? Donanım emulasyonu gerekli mi?

Uygulama Değerlendirmesinde Kullanılan Faktörler



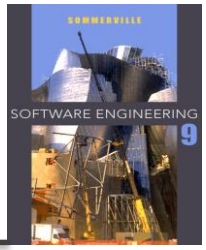
Faktör	Sorular
Anlaşılabilirlik	Mevcut sistemin kaynak kodunu anlamak ne kadar zor? Kullanılan kontrol yapıları ne kadar karmaşık? Değişkenlerin işlevlerini yansıtan anlamlı isimleri var mı?
Dokümantasyon	Hangi sistem belgeleri mevcut? Dokümantasyon eksiksiz, tutarlı ve güncel mi?
Veri	Sistem için açık bir veri modeli var mı? Veriler dosyalar arasında ne ölçüde çoğaltılır? Sistem tarafından kullanılan veriler güncel ve tutarlı mı?
Verim	Uygulamanın performansı yeterli mi? Performans sorunlarının sistem kullanıcıları üzerinde önemli bir etkisi var mı?

Uygulama Değerlendirmesinde Kullanılan Faktörler



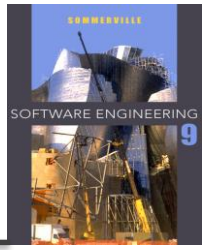
Faktör	Sorular
Programlama dili	Sistemi geliştirmek için kullanılan programlama dili için modern derleyiciler mevcut mu? Yeni sistem geliştirme için programlama dili hala kullanılıyor mu?
Konfigürasyon yönetimi	Sistemin tüm parçalarının tüm sürümleri bir konfigürasyon yönetim sistemi tarafından yönetiliyor mu? Mevcut sistemde kullanılan bileşenlerin sürümlerinin açık bir açıklaması var mı?
Test verisi	Sistem için test verileri mevcut mu? Sisteme yeni özellikler eklendiğinde gerçekleştirilen regresyon testlerinin bir kaydı var mı?
Personel becerileri	Uygulamayı sürdürme becerisine sahip insanlar var mı? Sistemde tecrübesi olan insanlar var mı?

Sistem Ölçümü



- Uygulama sisteminin kalitesinin bir değerlendirmesini yapmak için nicel veri toplayabilirsiniz.
  - Sistem değişikliği isteklerinin sayısı;
  - Sistem tarafından kullanılan farklı kullanıcı arayüzlerinin sayısı;
  - Sistem tarafından kullanılan veri hacmi.

Bölüm 2'nin Anahtar Noktaları



- 3 tür yazılım bakımı vardır: hata düzeltme, yazılımı yeni bir ortamda çalışacak şekilde değiştirme ve yeni veya değiştirilmiş gereksinimleri uygulama.
- Yazılım yeniden mühendisliği, yazılımın anlaşılmasını ve değiştirilmesini kolaylaştırmak için yeniden yapılandırma ve yeniden belgeleme ile ilgilidir.
- Yeniden düzenleme, işlevselliği koruyan program değişiklikleri yapma, önleyici bir bakım biçimidir.
- Eski bir sistemin iş değeri ve uygulamanın kalitesi, bir sistemin değiştirilmesi, dönüştürülmesi veya sürdürülmesi gerekip gerekmediğine karar vermeye yardımcı olmak için değerlendirilmelidir.