IT522 – Yazılım Mühendisliği 2021

PhD Furkan Gözükara, Toros University

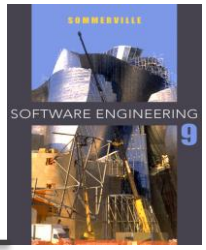
https://github.com/FurkanGozukara/Yazilim-Muhendisligi-IT522-2021

Ders 2

Yazılım Süreçleri

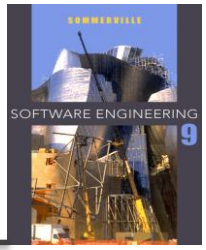
Kaynak : https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Presentations/index.html

Ders 2’de İşlenen Konular



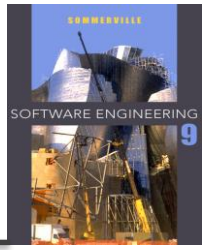
- ✧ Yazılım süreç modelleri
- ✧ Süreç faaliyetleri
- ✧ Değişimle başa çıkmak
- ✧ Rasyonel Birleşik Süreç
  - RUP, genellikle nesne ve / veya bileşen tabanlı teknolojilere dayalı sistemler geliştirmek için kullanılan, kuralcı, iyi tanımlanmış bir sistem geliştirme sürecidir.
  - Yazılım geliştirmeye yinelemeli, gereksinimlere dayalı ve mimari merkezli bir yaklaşım benimsemek gibi sağlam yazılım mühendisliği ilkelerine dayanmaktadır.
  - Modern bir yazılım sürecine bir örnek.

Yazılım Süreçleri



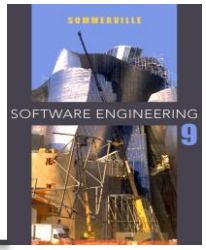
- Bir yazılım sistemi geliştirmek için gerekli olan yapılandırılmış bir dizi faaliyet.
- Birçok farklı yazılım süreci vardır ancak hepsi şunları içerir:
  - Spesifikasyon - sistemin ne yapması gerektiğini tanımlamak;
  - Tasarım ve uygulama - sistemin organizasyonunu tanımlamak ve sistemi uygulamak;
  - Doğrulama - müşterinin istediğini yaptığını kontrol etmek;
  - Değişim - değişen müşteri ihtiyaçlarına yanıt olarak sistemi değiştirmek.
- Bir yazılım süreci modeli, bir sürecin soyut bir temsilidir. Belirli bir perspektiften bir sürecin açıklamasını sunar.

Yazılım Süreci Açıklamaları



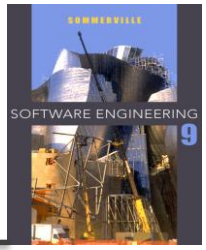
- Süreçleri açıklarken ve tartışırken genellikle bu süreçlerdeki bir veri modeli belirleme, bir kullanıcı arayüzü tasarlama vb. faaliyetlerden ve bu faaliyetlerin sıralanmasından bahsediyoruz.
- Süreç açıklamaları şunları da içerebilir:
  - Bir süreç faaliyetinin sonucu olan ürünler;
  - Sürece dahil olan kişilerin sorumluluklarını yansıtan roller;
  - Bir proses aktivitesinin yürürlüğe girmesinden veya bir ürünün üretilmesinden önce ve sonra doğru olan ön ve son koşullar.

Plan Odaklı Ve Çevik Süreçler



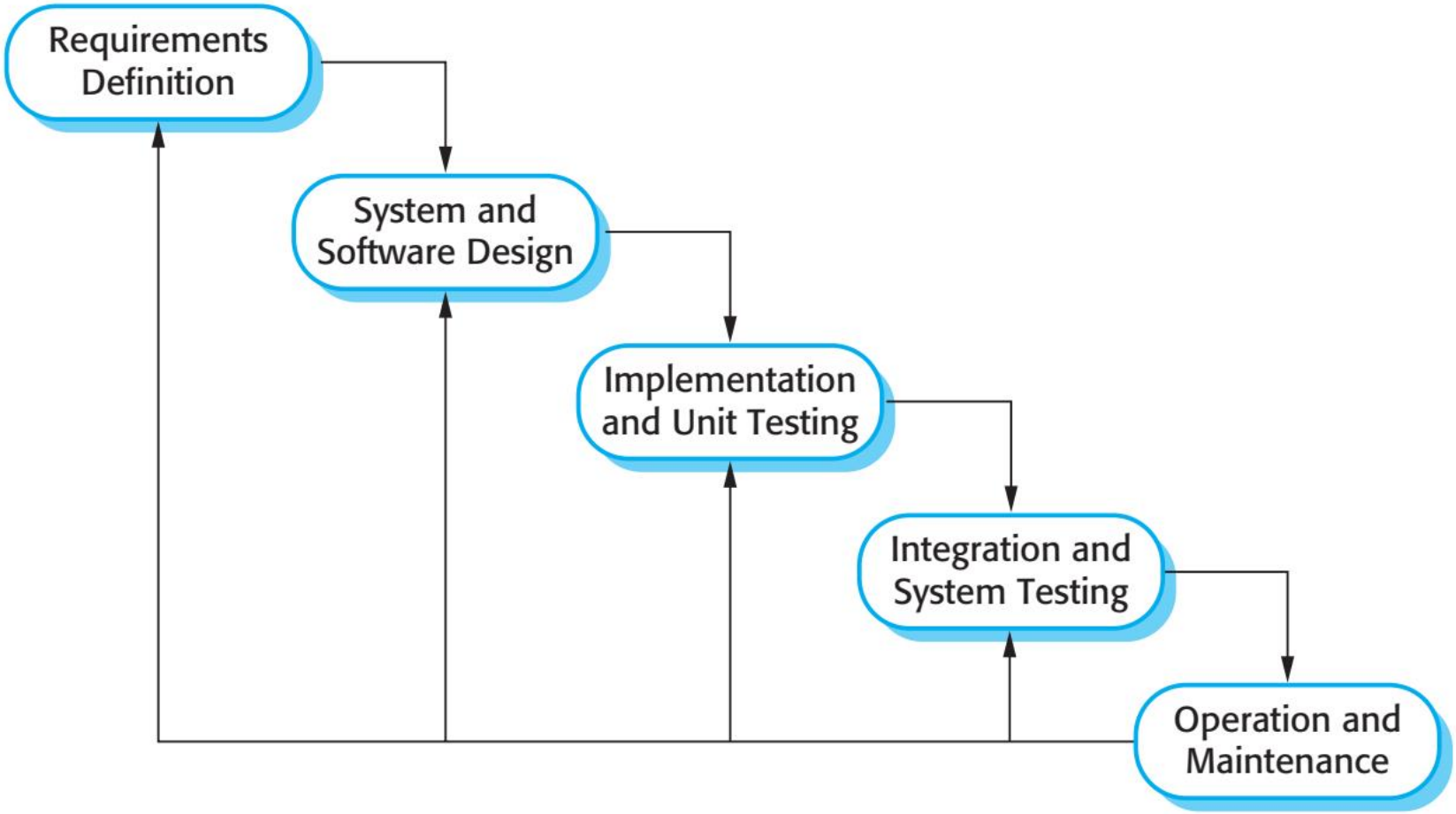
- Plan odaklı süreçler, tüm süreç faaliyetlerinin önceden planlandığı ve ilerlemenin bu plana göre ölçüldüğü süreçlerdir.
- Çevik süreçlerde planlama artımlıdır ve süreci değişen müşteri gereksinimlerini yansıtacak şekilde değiştirmek daha kolaydır.
- Pratikte çoğu pratik süreç, hem plan odaklı hem de çevik yaklaşımların unsurlarını içerir.
- Doğru veya yanlış yazılım süreçleri yoktur. Yani her yazılıma özel süreç olabilir.

Yazılım Süreçlerinin Modelleri

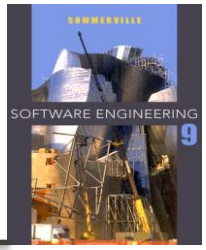


- ✧ Şelale modeli
  - Plan odaklı model. Spesifikasyon ve geliştirmenin ayrı ve farklı aşamaları vardır.
- ✧ Artımlı geliştirme
  - Spesifikasyon, geliştirme ve doğrulama serpiştirilmiştir. Plan odaklı veya çevik olabilir.
- ✧ Yeniden kullanım odaklı yazılım mühendisliği
  - Sistem, mevcut bileşenlerden bir araya getirilmiştir. Plan odaklı veya çevik olabilir.
- ✧ Uygulamada, çoğu büyük sistem, tüm bu modellerden öğeleri içeren bir süreç kullanılarak geliştirilir.

Şelale Modeli

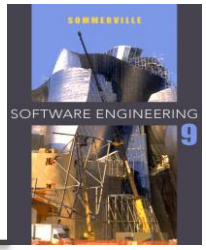


Şelale Model Aşamaları



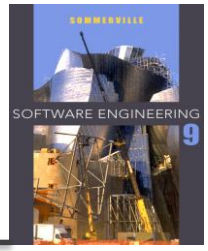
- Şelale modelinde ayrı tanımlanmış aşamalar vardır:
  - Gereksinim analizi ve tanımı
  - Sistem ve yazılım tasarımı
  - Uygulama ve birim testi
  - Entegrasyon ve sistem testi
  - Operasyon ve bakım
- Şelale modelinin temel dezavantajı, sürece başladıktan sonra süreci değiştirmenin zor olmasıdır. Prensip olarak, bir sonraki aşamaya geçmeden önce aşamanın tamamlanması gerekir.

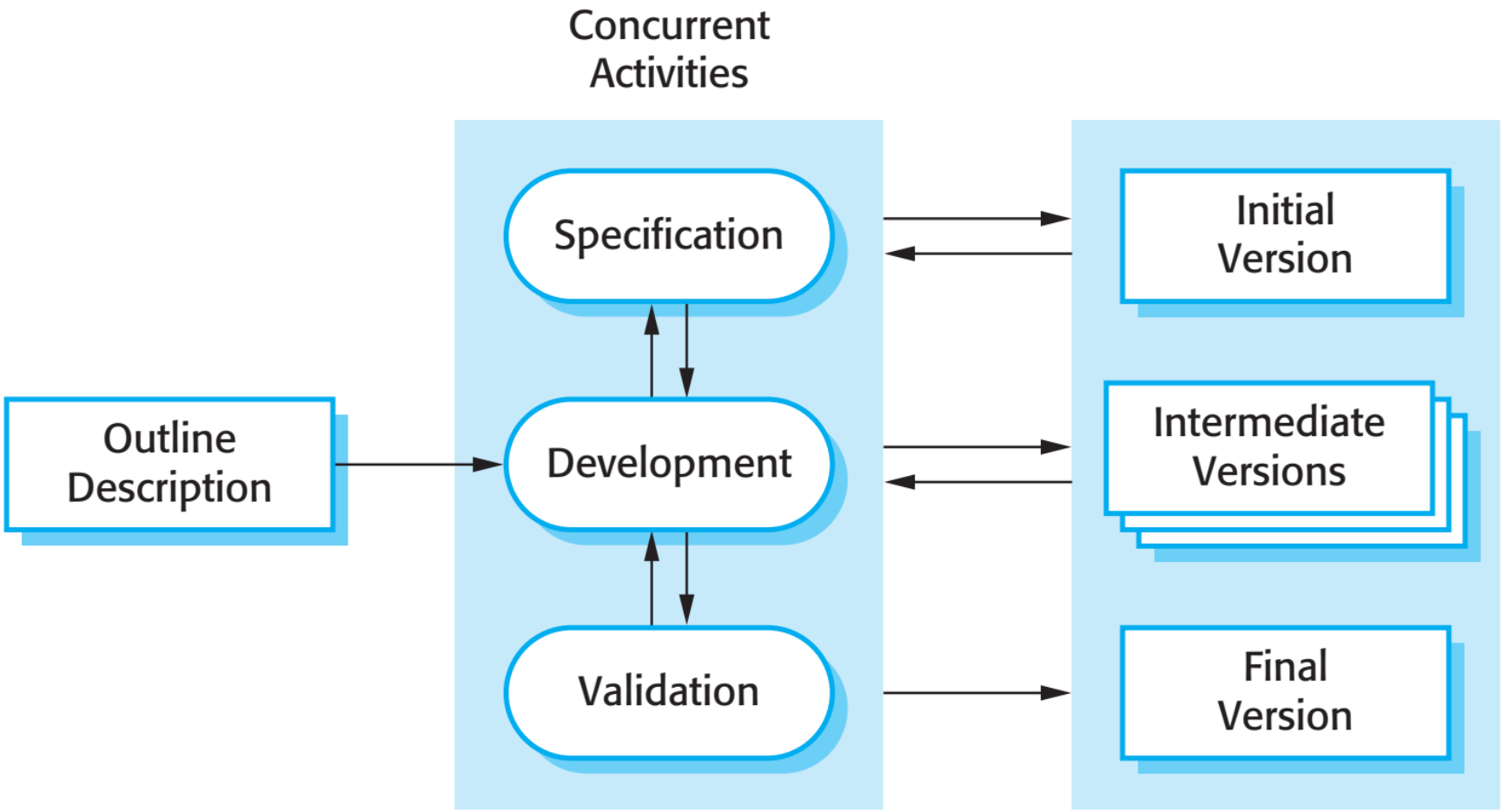
Şelale Modeli Problemleri



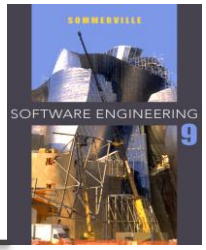
- Projenin farklı aşamalara esnek olmayan şekilde bölümlenmesi, değişen müşteri gereksinimlerine yanıt vermeyi zorlaştırır.
  - Bu nedenle, bu model yalnızca gereksinimler iyi anlaşıldığında ve tasarım sürecinde değişiklikler oldukça sınırlı olduğunda uygundur.
  - Çok az iş sisteminin istikrarlı gereksinimleri vardır.
- Şelale modeli, çoğunlukla bir sistemin birkaç sahada geliştirildiği büyük sistem mühendisliği projeleri için kullanılır.
  - Bu durumlarda, şelale modelinin plan odaklı yapısı, çalışmayı koordine etmeye yardımcı olur.

Artımlı Geliştirme



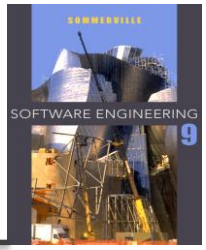


Artımlı Geliştirme Faydaları



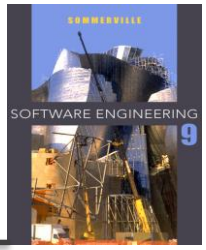
- Değişen müşteri gereksinimlerini karşılamanın maliyeti azaltılır.
  - Yeniden yapılması gereken analiz ve dokümantasyon miktarı, şelale modelinde gerekenden çok daha azdır.
- Yapılan geliştirme çalışmaları hakkında müşteri geri bildirimi almak daha kolaydır.
  - Müşteriler, yazılımın gösterimleri hakkında yorum yapabilir ve ne kadarının uygulandığını görebilir.
- Yararlı yazılımın müşteriye daha hızlı teslimi ve dağıtımı mümkündür.
  - Müşteriler, bir şelale süreciyle mümkün olandan daha erken bir zamanda yazılımı kullanabilir ve ondan değer kazanabilir.

Artımlı Geliştirme Sorunları



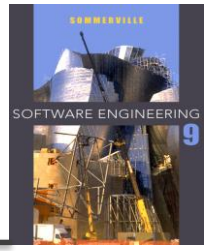
- Süreç görünmez.
  - Yöneticilerin ilerlemeyi ölçmek için düzenli çıktılara ihtiyacı vardır. Sistemler hızlı bir şekilde geliştirilirse, sistemin her sürümünü yansıtan belgeler üretmek uygun maliyetli değildir.
- Sistem yapısı, yeni artışlar eklendikçe bozulma eğilimindedir .
  - Yazılımı iyileştirmek için yeniden düzenleme için zaman ve para harcanmadıkça, düzenli değişim yapısını bozma eğilimindedir. Daha fazla yazılım değişikliğini dahil etmek giderek daha zor ve maliyetli hale gelir.

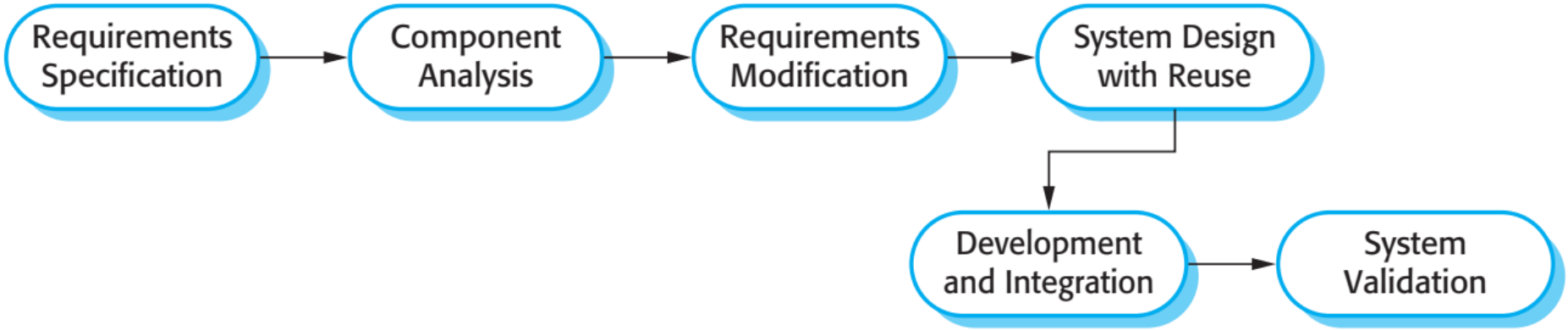
Yeniden Kullanım Odaklı Yazılım Mühendisliği



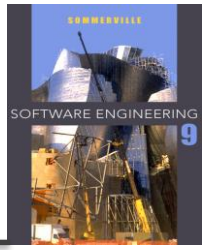
- Sistemlerin mevcut bileşenlerden veya COTS (Commercial-off-the-shelf) (ticari satışa hazır) sistemlerden entegre edildiği sistematik yeniden kullanıma dayanır.
- Süreç aşamaları
  - Bileşen analizi;
  - Gereksinim değişikliği;
  - Yeniden kullanım ile sistem tasarımı;
  - Geliştirme ve entegrasyon.
- Yeniden kullanım artık birçok türde iş sistemi oluşturmak için standart yaklaşımdır
  - Yeniden kullanım 16. Bölümde daha ayrıntılı olarak anlatılmıştır.

Yeniden Kullanım Odaklı Yazılım Mühendisliği



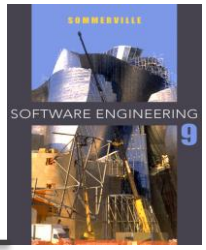


Yazılım Bileşeni Türleri



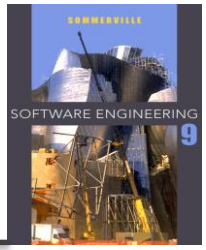
- Hizmet standartlarına göre geliştirilmiş ve uzaktan çağrılarak kullanılabilen web servisleri.
- .NET veya J2EE gibi bir bileşen çerçevesi ile entegre edilecek bir paket olarak geliştirilen nesne koleksiyonları.
- Belirli bir ortamda kullanılmak üzere yapılandırılmış bağımsız yazılım sistemleri (COTS).

Süreç Aktivitileri



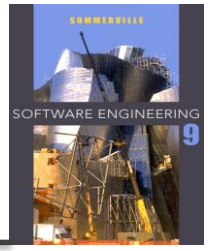
- Gerçek yazılım süreçleri, bir yazılım sistemini belirleme, tasarlama, uygulama ve test etme genel amacı ile teknik, işbirliğine dayalı ve yönetsel faaliyetlerin aralıklı dizilmiş dizileridir.
- Spesifikasyon, geliştirme, doğrulama ve değişimden oluşan dört temel süreç etkinliği, farklı geliştirme süreçlerinde farklı şekilde düzenlenir. Şelale modelinde, sırayla düzenlenirlerken artımlı gelişimde aralıklıdırlar.

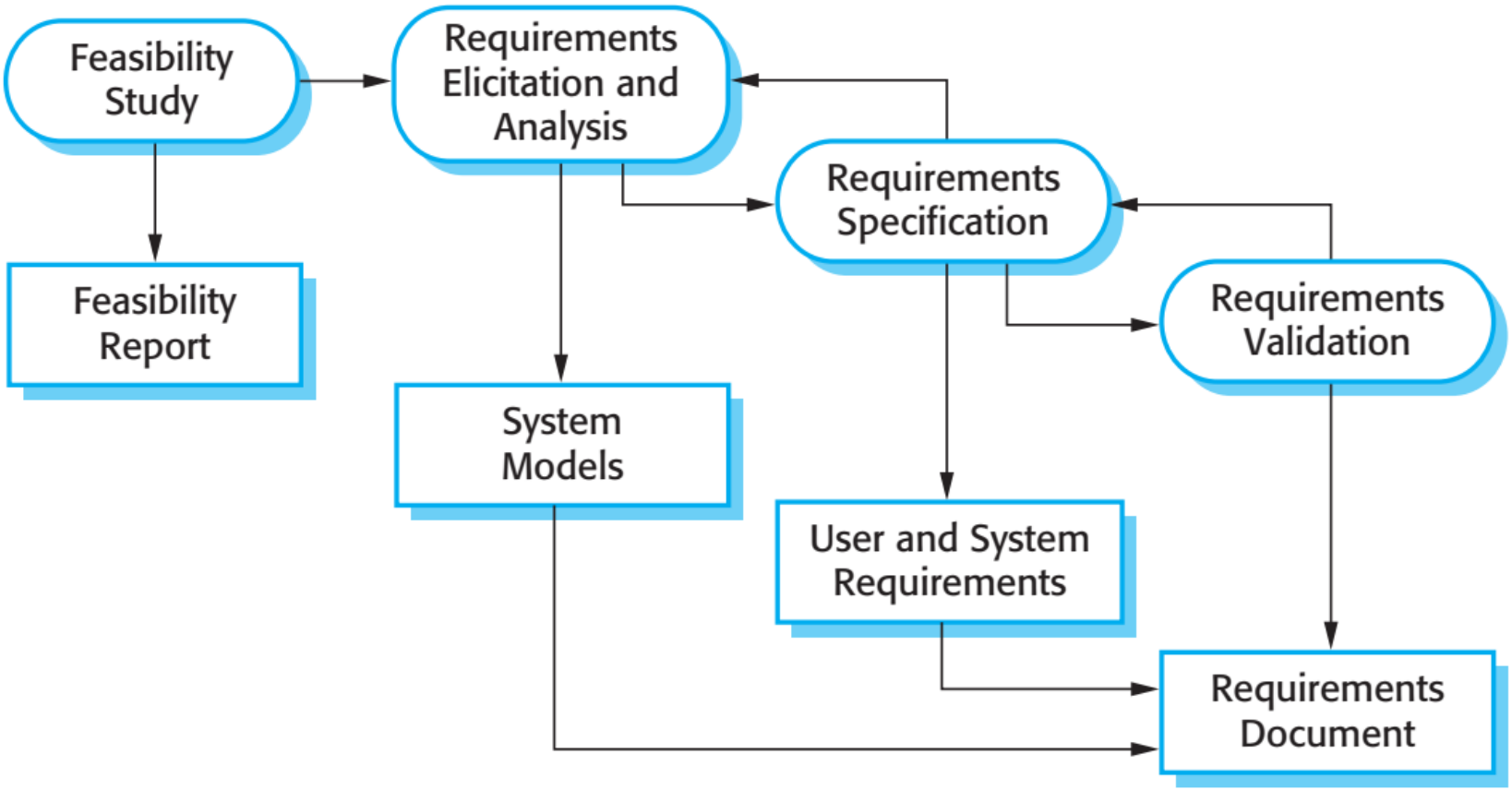
Yazılım Özellikleri



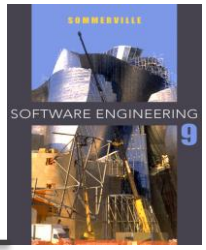
- Hangi hizmetlerin gerekli olduğu ve sistemin işleyişi ve gelişimi üzerindeki kısıtlamalara karar verme süreci.
- Gereksinimler mühendisliği süreci
  - Fizibilite çalışması
    - Sistemi kurmak teknik ve mali olarak uygun mu?
  - Gereksinimlerin ortaya çıkarılması ve analizi
    - Sistem paydaşları sistemden ne ister veya ne bekler?
  - Gereksinim özellikleri
    - Gereksinimlerin ayrıntılı olarak tanımlanması
  - Gereksinimlerin doğrulanması
    - Gereksinimlerin geçerliliğini kontrol etmek

Gereksinim Mühendisliği Süreci



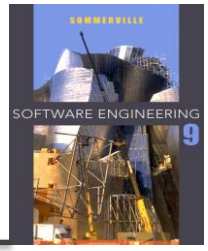


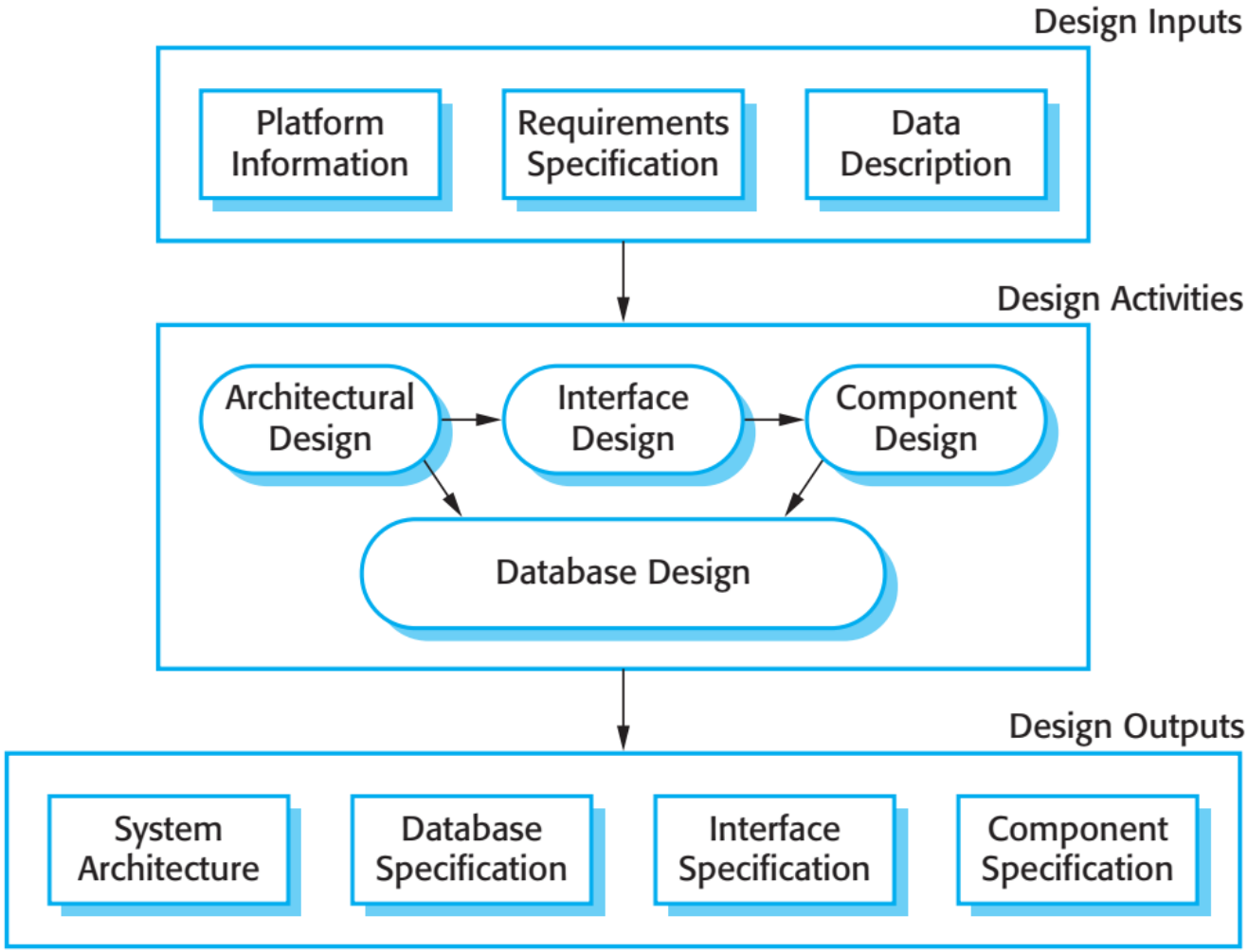
Yazılım Tasarımı Ve Entegrasyonu



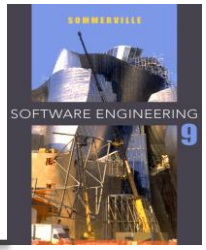
- Sistem spesifikasyonunu çalıştırılabilir bir sisteme dönüştürme süreci.
- Yazılım Tasarımı
  - Spesifikasyonu gerçekleştiren bir yazılım yapısı tasarlayın;
- Entegrasyon
  - Bu yapıyı çalıştırılabilir bir programa çevirin;
- Tasarım ve entegrasyon faaliyetleri yakından ilişkilidir ve birbirleri ile iç içe olabilirler.

Tasarım Sürecinin Genel Bir Modeli



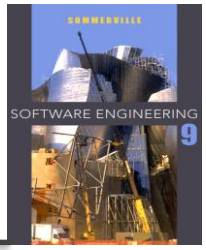


Tasarım Faaliyetleri



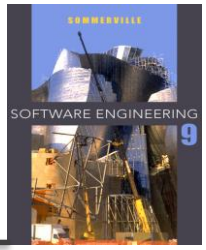
- Sistemin genel yapısını, ana bileşenleri (bazen alt sistemler veya modüller olarak adlandırılır), bunların ilişkilerini ve nasıl dağıtıldıklarını belirlediğimiz ***Mimari Tasarım***.
- Sistem bileşenleri arasındaki arayüzleri tanımladığınız ***Arayüz Tasarımı***.
- ***Bileşen Tasarımı***, her sistem bileşenini alıp nasıl çalışacağı tasarladığımız süreç.
- Sistem veri yapılarını ve bunların bir veritabanında nasıl temsil edileceği tasarladığımız ***Veritabanı Tasarımı***.

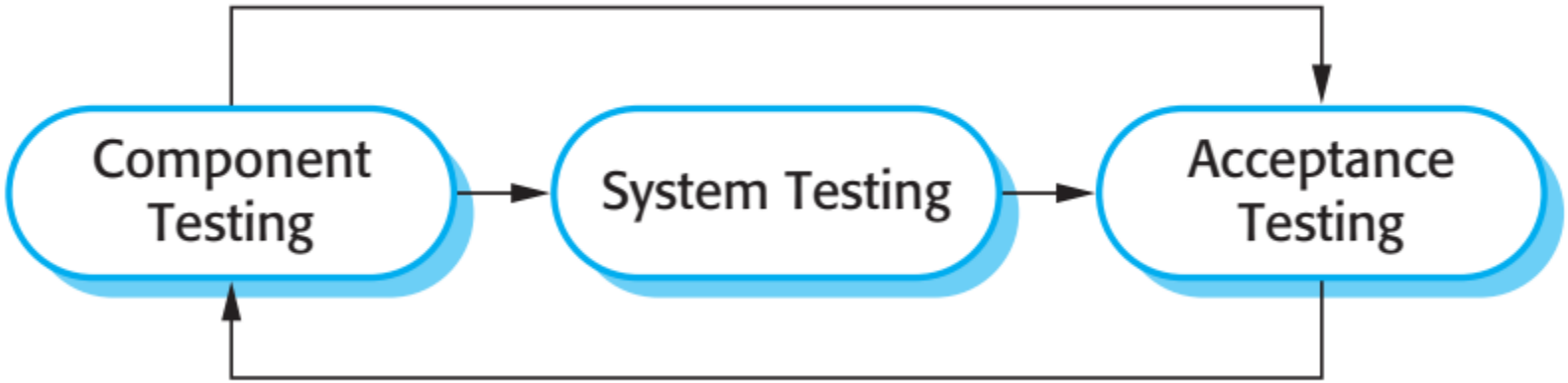
Yazılım Doğrulama



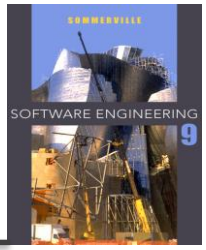
- Tasdikleme ve doğrulama (Verification and Validation) (V & V), bir sistemin spesifikasyonuna uygun olduğunu ve sistem müşterisinin gereksinimlerini karşıladığını göstermeyi amaçlar.
- Kontrol ve inceleme süreçlerini ve sistem testini içerir.
- Sistem testi, sistem tarafından işlenecek gerçek verilerin spesifikasyonundan türetilen test senaryoları ile sistemin yürütülmesini içerir.
- Test, en yaygın kullanılan V & V etkinliğidir.

Test Aşamaları



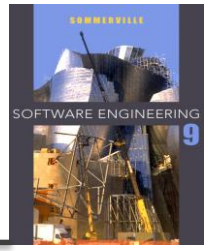


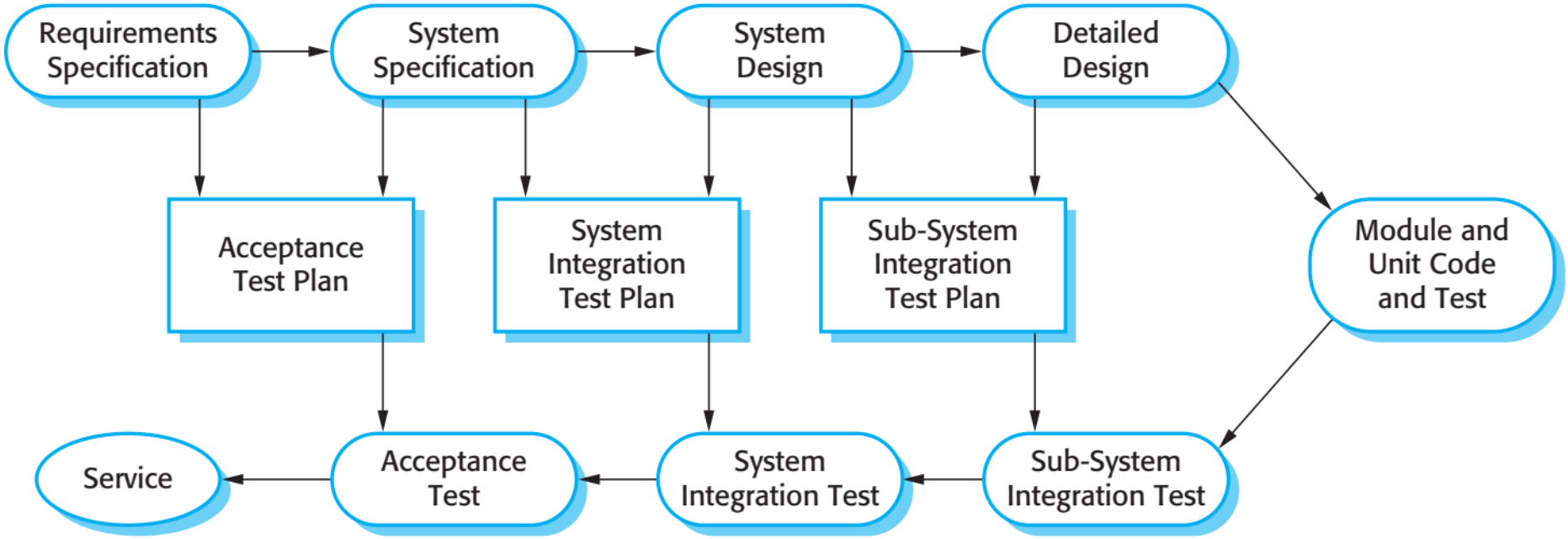
Test Aşamaları



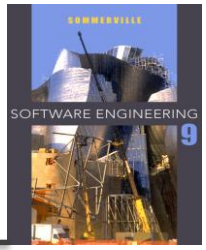
- ✧ Geliştirme veya bileşen testi
  - Bireysel bileşenler bağımsız olarak test edilir;
  - Bileşenler, bu varlıkların fonksiyonları veya nesneleri veya tutarlı gruplamaları olabilir.
- ✧ Sistem testi
  - Sistemin bir bütün olarak test edilmesi. Ortaya çıkan özelliklerin test edilmesi özellikle önemlidir.
- ✧ Kabul testleri
  - Sistemin müşterinin ihtiyaçlarını karşılayıp karşılamadığını kontrol etmek için müşteri verileriyle test etme.

Plan Odaklı Bir Yazılım Sürecindeki Test Aşamaları



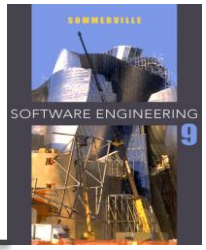


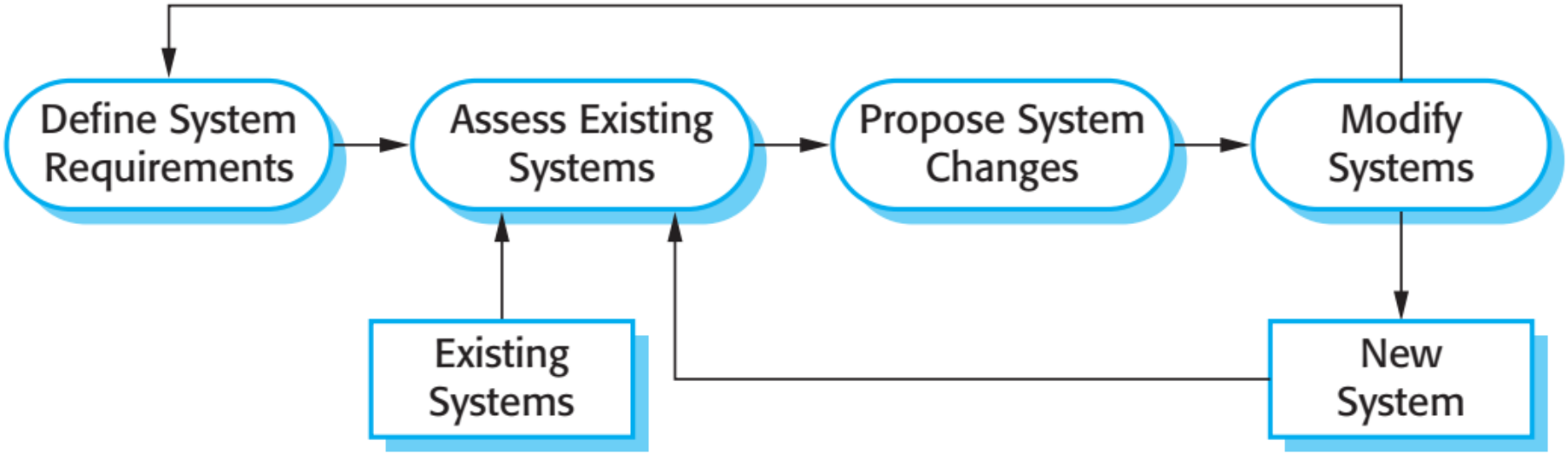
Yazılım Değişimi



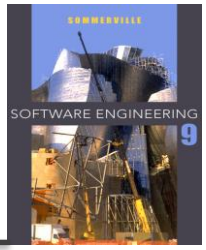
- Yazılım doğası gereği esnektir ve değişebilir.
- Değişen iş koşulları nedeniyle gereksinimler değiştikçe, işi destekleyen yazılımın da gelişmesi ve değişmesi gerekir.
- Geliştirme ve değişim (bakım) arasında eskiden bir sınır çizilmesine rağmen, giderek daha az sayıda sistem tamamen yeni olduğu için bu giderek önemsiz hale gelmektedir.

Sistem Değişimi



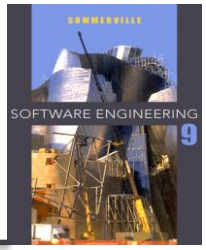


Bölüm 1 Anahtar Noktalar

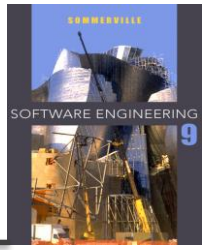


- Yazılım süreçleri, bir yazılım sisteminin üretilmesiyle ilgili faaliyetlerdir. Yazılım süreç modelleri, bu süreçlerin soyut temsilleridir.
- Genel süreç modelleri, yazılım süreçlerinin organizasyonunu tanımlar. Bu genel modellerin örnekleri arasında 'şelale' modeli, aşamalı geliştirme ve yeniden kullanıma yönelik geliştirme modelleri bulunmaktadır.

Bölüm 1 Anahtar Noktalar



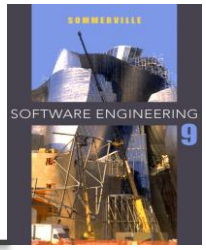
- Gereksinim mühendisliği, bir yazılım spesifikasyonu geliştirme sürecidir.
- Tasarım ve uygulama süreçleri, bir gereksinim spesifikasyonunu yürütülebilir bir yazılım sistemine dönüştürmekle ilgilidir.
- Yazılım doğrulama, sistemin özelliklerine uygunluğunun ve sistem kullanıcılarının gerçek ihtiyaçlarını karşıladığının kontrol edilmesi sürecidir.
- Yazılım değişimi, mevcut yazılım sistemlerini yeni gereksinimleri karşılayacak şekilde değiştirdiğinizde gerçekleşir. Yazılım, kullanışlı kalması için gelişmelidir.



Ders 2 - Yazılım Süreçleri

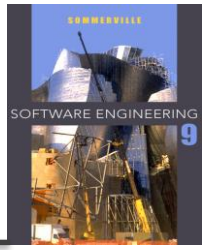
Bölüm 2

Değişimle Başa Çıkmak



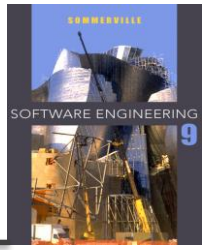
- Tüm büyük yazılım projelerinde değişim kaçınılmazdır.
  - İş değişiklikleri, yeni ve değişen sistem gereksinimlerine yol açar
  - Yeni teknolojiler, uygulamaları iyileştirmek için yeni olanaklar sunar
  - Değişen platformlar, uygulama değişiklikleri gerektirir
- Değişiklik, yeniden çalışmaya yol açar, bu nedenle değişim maliyetleri hem yeniden çalışmayı (örneğin, gereksinimleri yeniden analiz etme) hem de yeni işlevsellik entegrasyon maliyetlerini içerir.

Yeniden İşleme Maliyetlerini Düşürmek



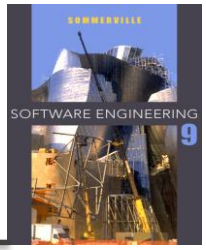
- **Değişimden kaçınma:** Yazılımın yeniden düzenlenmesi gerekmeden önce olası değişiklikleri öngörebilen etkinlikleri içeren yazılım süreci.
  - Örneğin, sistemin bazı temel özelliklerini müşterilere göstermek için bir prototip sistemi geliştirilebilir.
- **Değişim toleransı:** değişikliklerin nispeten düşük maliyetle gerçekleştirilebilmesi tasarım aşamasında tolerans sağlayın.
  - Bu normalde bir tür artımlı geliştirme içerir. Önerilen değişiklikler, henüz geliştirilmemiş artışlarla uygulanabilir. Bu imkansızsa, değişikliği dahil etmek için yalnızca tek bir artış (sistemin küçük bir kısmı) değiştirilebilir.

Yazılım Prototipleme



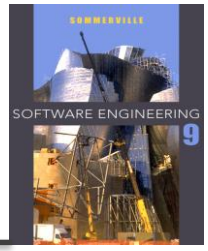
- Bir prototip, konseptleri göstermek ve tasarım seçeneklerini denemek için kullanılan bir sistemin ilk sürümüdür.
- Bir prototip şunlarda kullanılabilir:
  - Gereksinimlerin ortaya çıkarılması ve doğrulanmasına yardımcı olmak için gereksinim mühendisliği süreci;
  - Tasarım süreçlerinde seçenekleri keşfetmek ve bir UI tasarımı geliştirmek için;
  - Test sürecinde arka arkaya testleri çalıştırmak için.

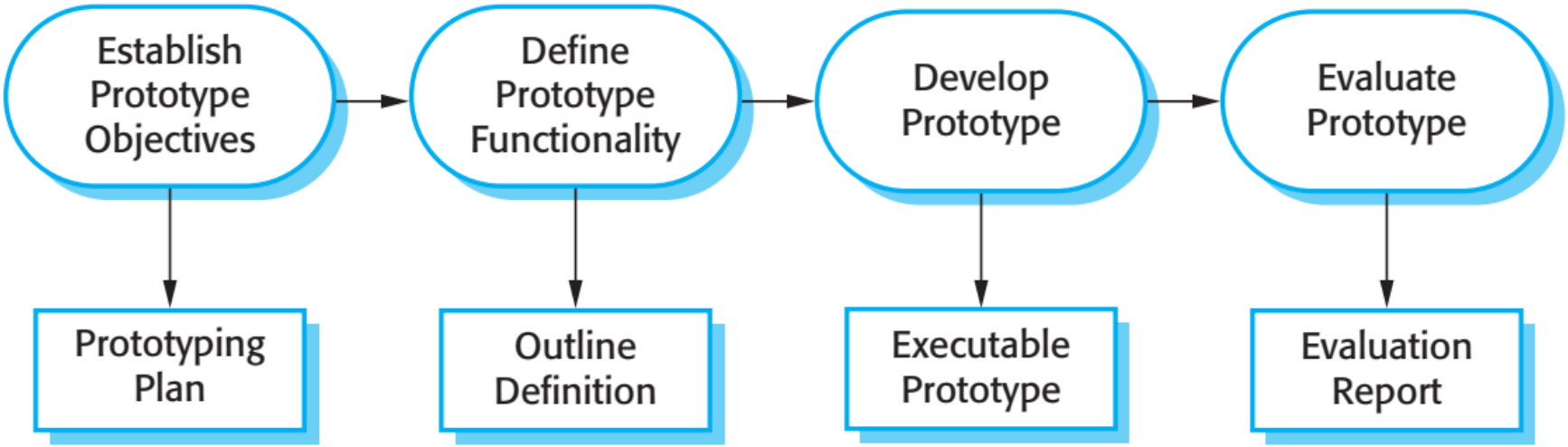
Prototip Oluşturmanın Faydaları



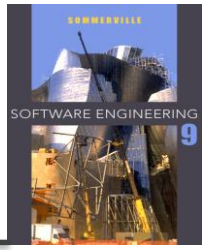
- İyileştirilmiş sistem kullanılabilirliği.
- Kullanıcıların gerçek ihtiyaçlarına daha yakın bir eşleşme.
- İyileştirilmiş tasarım kalitesi.
- İyileştirilmiş bakım kolaylığı.
- Azaltılmış geliştirme çabası.

Prototip Geliştirme Süreci



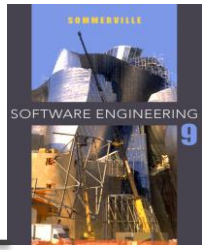


Prototip Geliştirme



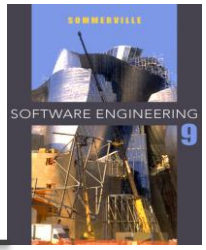
- Hızlı prototip oluşturma dillerine veya araçlarına dayalı olabilir
- Fonksiyonelliğin dışarıda bırakılmasını içerebilir
  - Prototip, ürünün iyi anlaşılmayan alanlarına odaklanmalıdır;
  - Prototipe hata denetimi ve kurtarma dahil edilmeyebilir;
  - Güvenilirlik ve güvenlik gibi işlevsel olmayan gereksinimler yerine işlevsel gereksinimlere odaklanın

Prototipler Atılmalıdır



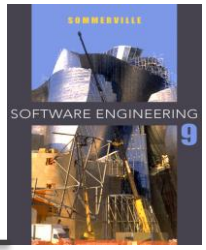
- Prototipler, bir üretim sistemi için iyi bir temel olmadığından geliştirildikten sonra atılmalıdır:
  - İşlevsel olmayan gereksinimleri karşılamak için sistemi ayarlamak imkansız olabilir;
  - Prototipler normalde belgelenmez;
  - Prototip yapısı genellikle hızlı değişimle bozulur;
  - Prototip muhtemelen normal kurumsal kalite standartlarını karşılamayacaktır.

Artımlı Teslimat



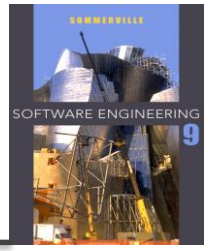
- Sistemi tek bir teslimat olarak sunmak yerine, geliştirme ve teslimat, gerekli işlevselliğin bir bölümünü sağlayan her bir artımla birlikte aşamalara bölünür.
- Kullanıcı gereksinimlerine öncelik verilir ve en yüksek öncelik gereksinimleri erken artışlara dahil edilir.
- Bir artımın geliştirilmesine başlandıktan sonra, gereksinimler dondurulur, ancak sonraki artışlar için gereksinimler gelişmeye devam edebilir.

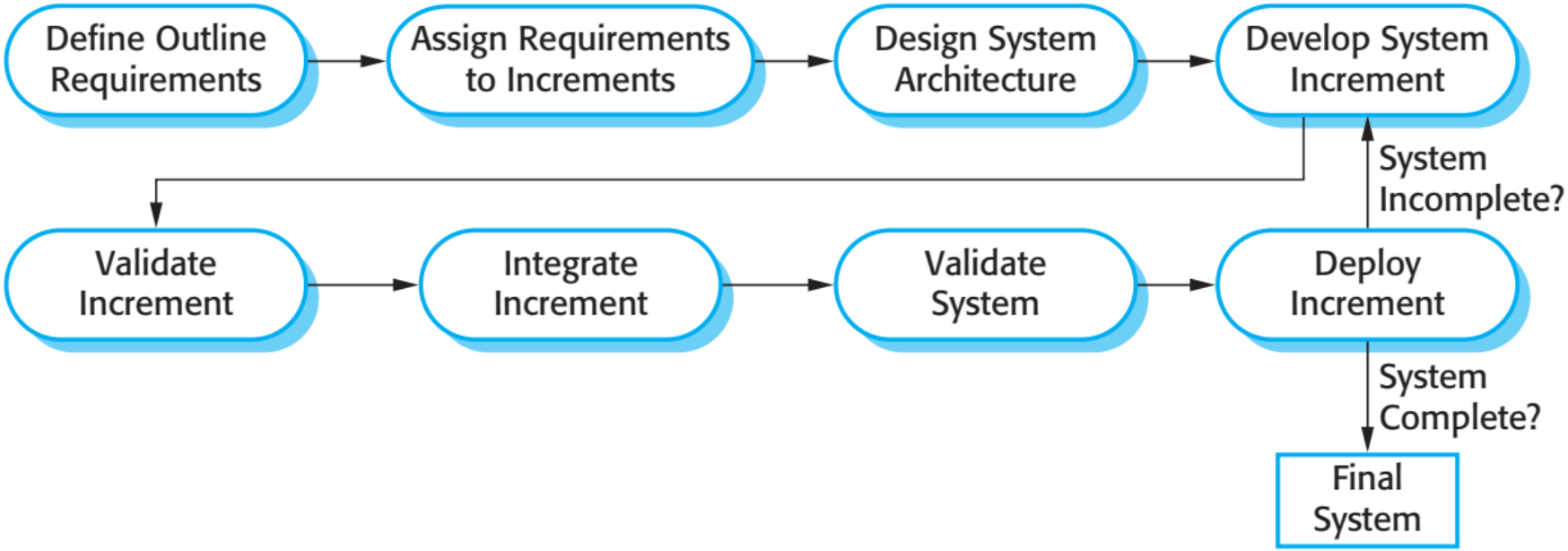
Artımlı Geliştirme Ve Teslimat



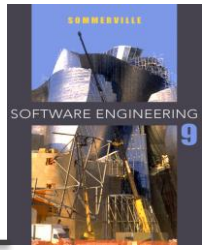
- Artımlı geliştirme
  - Sistemi aşamalar halinde geliştirin ve bir sonraki artışın geliştirilmesine geçmeden önce her bir artışı değerlendirin;
  - Çevik yöntemlerde kullanılan normal yaklaşım;
  - Kullanıcı / müşteri vekili tarafından yapılan değerlendirme.
- Artımlı teslimat
  - Son kullanıcılar tarafından kullanılmak üzere bir artım dağıtın;
  - Yazılımın pratik kullanımı hakkında daha gerçekçi değerlendirme;
  - Arttırmalar, değiştirilen sistemden daha az işlevselliğe sahip olduğundan, değiştirme sistemleri için uygulanması zordur.

Artımlı Teslimat



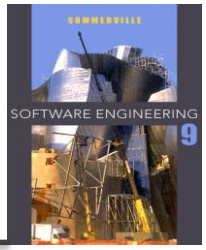


Artımlı Dağıtım Avantajları



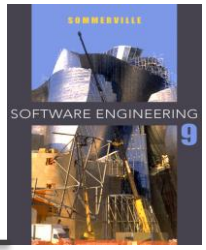
- Müşteri istekleri, her bir artışla sunulabilir, böylece sistem işlevselliği daha erken kullanılabilir hale gelir.
- Erken artışlar, sonraki artışlar için gereksinimleri ortaya çıkarmaya yardımcı olacak bir prototip görevi görür.
- Daha düşük genel proje başarısızlığı riski.
- En yüksek öncelikli sistem hizmetleri en çok testi alma eğilimindedir.

Artımlı Dağıtım Sorunları



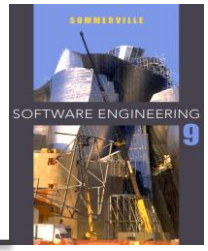
- Çoğu sistem, sistemin farklı bölümleri tarafından kullanılan bir dizi temel tesis gerektirir.
  - İhtiyaçlar, bir artış uygulanana kadar ayrıntılı olarak tanımlanmadığından, tüm artışların ihtiyaç duyduğu ortak tesisleri belirlemek zor olabilir.
- Yinelemeli süreçlerin özü, spesifikasyonun yazılımla birlikte geliştirilmesidir.
  - Ancak, bu, tüm sistem spesifikasyonunun sistem geliştirme sözleşmesinin bir parçası olduğu birçok kuruluşun satın alma modeliyle çelişir.

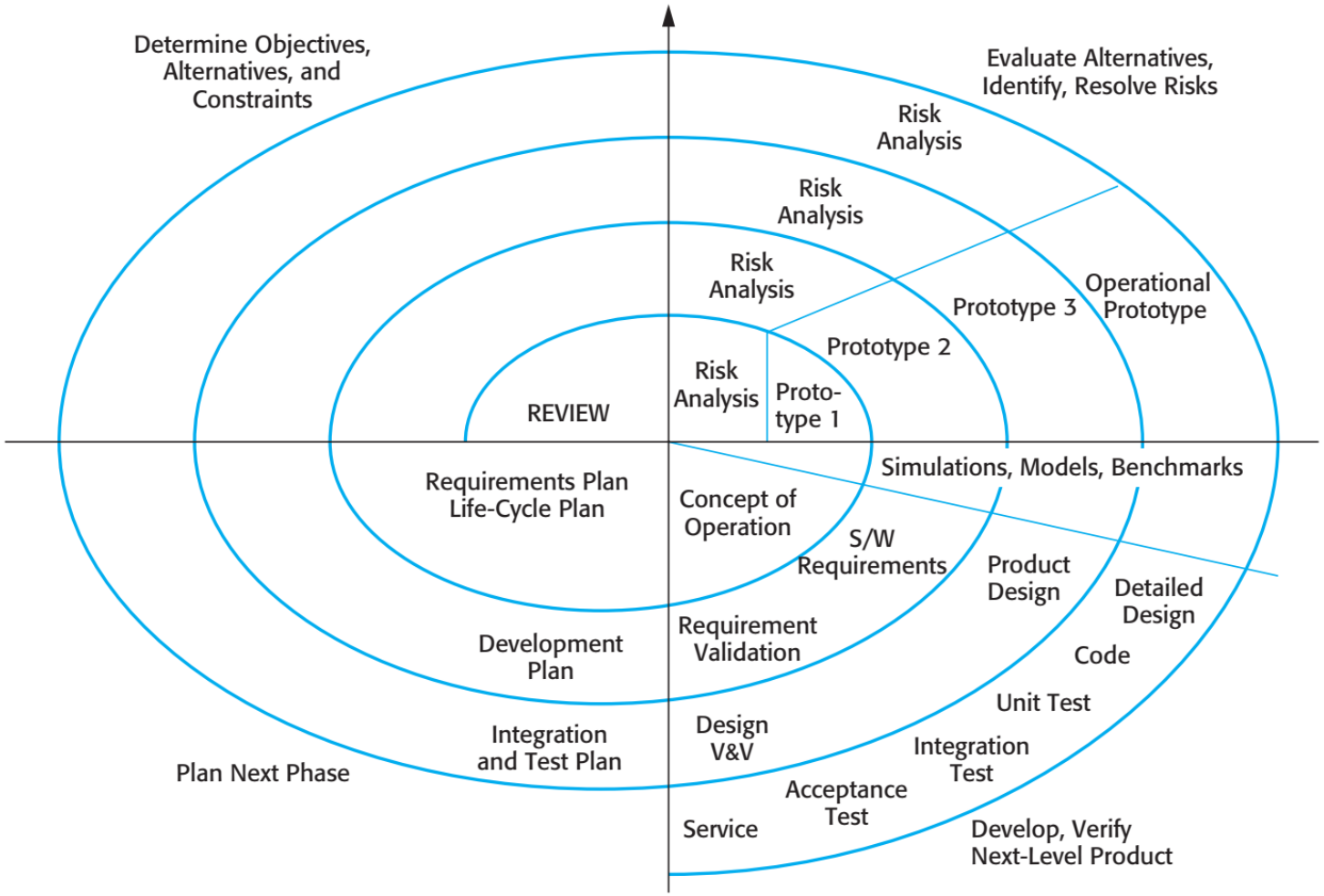
Boehm'in Spiral Modeli



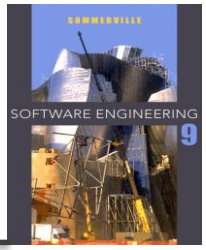
- Süreç, geriye dönük izleme içeren bir faaliyetler dizisi yerine bir sarmal olarak temsil edilir.
- Spiraldeki her döngü, süreçteki bir aşamayı temsil eder.
- Spesifikasyon veya tasarım gibi sabit aşama yoktur: gerekli olana bağlı olarak spiraldeki döngüler seçilir.
- Süreç boyunca riskler açıkça değerlendirilir ve çözülür.

Boehm'in Yazılım Sürecinin Spiral Modeli



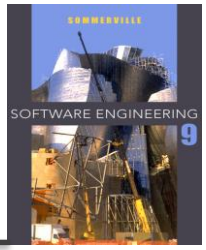


Spiral Model Sektörler



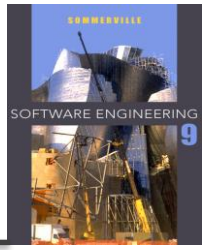
- Hedef belirleme
  - Aşama için özel hedefler belirlenir.
- Risk değerlendirmesi ve azaltma
  - Riskler değerlendirilir ve temel riskleri azaltmak için faaliyetler başlatılır.
- Geliştirme ve doğrulama
  - Sistem için genel modellerden herhangi biri olabilecek bir geliştirme modeli seçilir.
- Planlama
  - Proje gözden geçirilir ve spiralin bir sonraki aşaması planlanır.

Spiral Model Kullanımı



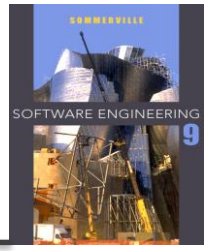
- Spiral model, insanların yazılım süreçlerinde yineleme hakkında düşünmelerine yardımcı olmada ve geliştirmeye risk odaklı yaklaşımı tanıtmada çok etkili olmuştur.
- Ancak pratikte, model pratik yazılım geliştirme için yayınlandığı şekliyle nadiren kullanılır.

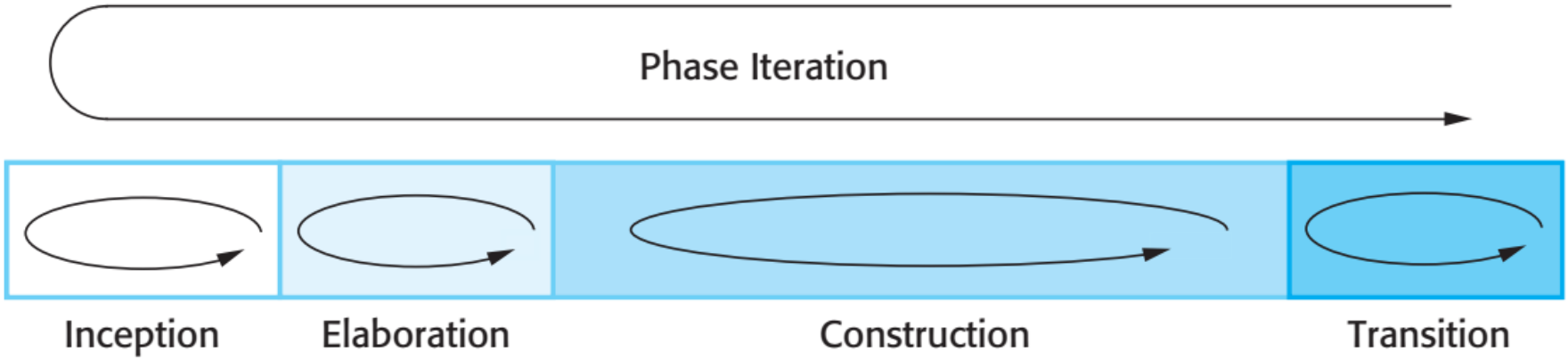
Rasyonel Birleşik Süreç



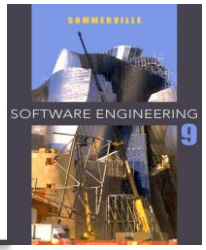
- UML ve ilgili süreç üzerindeki çalışmalardan türetilen modern bir genel süreç.
- Daha önce tartışılan 3 genel süreç modelinin yönlerini bir araya getirir.
- Normalde 3 perspektiften tanımlanmıştır
  - Zaman içindeki aşamaları gösteren dinamik bir bakış açısı;
  - Süreç etkinliklerini gösteren statik bir bakış açısı;
  - İyi uygulamayı öneren pratik bir bakış açısı.

Rasyonel Birleştirilmiş Süreçteki Aşamalar



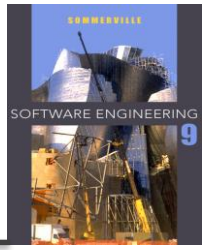


RUP Aşamaları



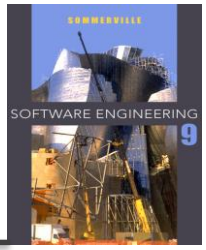
- ✧ Başlangıç
  - Sistem için iş senaryosu oluşturun.
- ✧ Detaylandırma
  - Sorun alanı ve sistem mimarisi hakkında bir anlayış geliştirin.
- ✧ İnşaat
  - Sistem tasarımı, programlama ve test.
- ✧ Geçiş
  - Sistemi işletim ortamında devreye alın.

RUP Yineleme



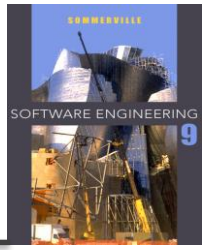
- Aşama içi yineleme
  - Her aşama, aşamalı olarak geliştirilen sonuçlarla yinelemelidir.
- Aşamalar arası yineleme
  - RUP modelindeki döngüde gösterildiği gibi, tüm fazlar kümesi artımlı olarak canlandırılabilir.

Rasyonel Birleştirilmiş Süreçteki Statik Iş Akışları



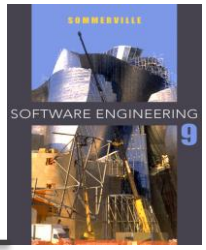
İş akışı	Açıklama
İş modelleme	İş süreçleri, iş kullanım durumları kullanılarak modellenir.
Gereksinimler	Sistemle etkileşime giren aktörler belirlenir ve sistem gereksinimlerini modellemek için kullanım senaryoları geliştirilir.
Analiz ve tasarım	Mimari modeller, bileşen modelleri, nesne modelleri ve sıra modelleri kullanılarak bir tasarım modeli oluşturulur ve belgelenir.
Uygulama	Sistemdeki bileşenler, uygulama alt sistemlerine dönüştürülür ve yapılandırılır. Tasarım modellerinden otomatik kod üretimi, bu süreci hızlandırmaya yardımcı olur.

Rasyonel Birleştirilmiş Süreçteki Statik Iş Akışları



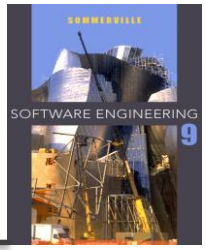
İş akışı	Açıklama
Test yapmak	Test, uygulama ile bağlantılı olarak gerçekleştirilen yinelemeli bir süreçtir. Sistem testi, uygulamanın tamamlanmasını takip eder.
Dağıtım	Bir ürün sürümü oluşturulur, kullanıcılara dağıtılır ve iş yerlerine yüklenir.
Yapılandırma ve değişiklik yönetimi	Bu destekleyici iş akışı, sistemdeki değişiklikleri yönetir (bkz. Ders 25).
Proje Yönetimi	Bu destekleyici iş akışı, sistem geliştirmeyi yönetir (bkz. Bölüm 22 ve 23).
Çevre	Bu iş akışı, uygun yazılım araçlarının yazılım geliştirme ekibinin kullanımına sunulmasıyla ilgilidir.

Rasyonel Birleştirilmiş Süreçteki Iyi Uygulama



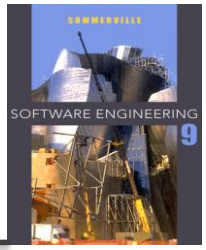
- ✧ Yazılımları yinelemeli olarak geliştirin
  - Müşteri önceliklerine göre artışları planlayın ve önce en yüksek öncelikli artışları sağlayın.
- ✧ Gereksinimleri yönetin
  - Müşteri gereksinimlerini açıkça belgeleyin ve bu gereksinimlerdeki değişiklikleri takip edin.
- ✧ Bileşen tabanlı mimariler kullanın
  - Sistem mimarisini bir dizi yeniden kullanılabilir bileşen olarak düzenleyin.

Rasyonel Birleştirilmiş Süreçteki Iyi Uygulama



- ✧ Görsel olarak model yazılım
  - Yazılımın statik ve dinamik görünümlerini sunmak için grafik UML modellerini kullanın.
- ✧ Yazılım kalitesini doğrulayın
  - Yazılımın kurumsal kalite standartlarını karşıladığından emin olun.
- ✧ Yazılımdaki değişiklikleri kontrol edin
  - Bir değişiklik yönetimi sistemi ve konfigürasyon yönetimi araçları kullanarak yazılım değişikliklerini yönetin.

Bölüm 2 Anahtar Noktaları



- Süreçler, değişimle baş etmeye yönelik faaliyetleri içermelidir. Bu, gereksinimler ve tasarım konusunda kötü kararlardan kaçınmaya yardımcı olan bir prototip oluşturma aşaması içerebilir.
- Süreçler yinelemeli geliştirme ve teslimat için yapılandırılabilir, böylece sistemi bir bütün olarak bozmadan değişiklikler yapılabilir.
- Rasyonel Birleştirilmiş Süreç, aşamalar halinde (başlangıç, detaylandırma, yapım ve geçiş) organize edilen ancak faaliyetleri (gereksinimler, analiz ve tasarım, vb.) bu aşamalardan ayıran modern bir genel süreç modelidir.