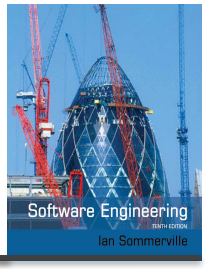
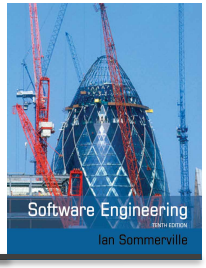


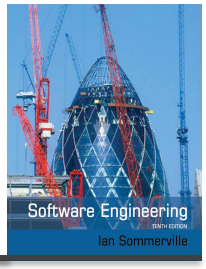
Yazılım Süreçleri

Yazılım Süreci



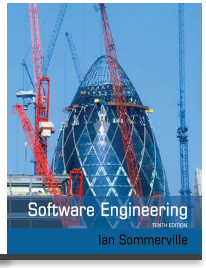
- Bir yazılım sistemi geliştirmek için gerekli olan yapılandırılmış faaliyetler dizisi.
- Birçok farklı yazılım süreci, ancak tümü şunları içerir:
  - Spesifikasyon – sistemin ne yapması gerektiğinin tanımlanması;
  - Tasarım ve Gerçekleştirim – sistemin organizasyonunun tanımlanması ve sistemin kodlanması;
  - Doğrulama – müşterinin istediğini yapıp yapmadığını kontrol etmek;
- Bir yazılım süreç modeli, bir sürecin soyut bir temsilidir. Belirli bir perspektiften bir sürecin tanımını sunar.

Yazılım süreç tanımları

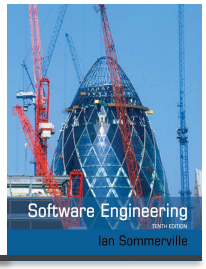


- Süreçleri tarif ederken ve tartışırken, genellikle bu süreçlerdeki veri modeli belirleme, kullanıcı arayüzü tasarlama vb. etkinliklerden ve bu etkinliklerin sıralanmasından bahsederiz.
- Süreç açıklamaları ayrıca şunları içerebilir:
  - Bir süreç faaliyetinin sonucu olan ürünler;
  - Sürece dahil olan kişilerin sorumluluklarını yansıtan roller;
  - Bir süreç faaliyetinin yürürlüğe girmesinden veya bir üründen önce ve sonra doğru olan ifadeler olan ön ve son koşullar.

Plan odaklı ve çevik süreçler

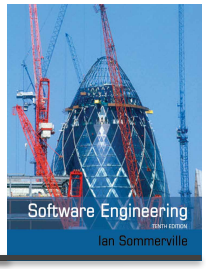


- Plan odaklı süreçler, tüm süreç aktivitelerinin önceden planlandığı ve bu plana göre ilerlemenin ölçüldüğü süreçlerdir.
- Çevik süreçlerde planlama aşamalıdır ve süreci değişen müşteri gereksinimlerini yansıtacak şekilde değiştirmek daha kolaydır.
- Uygulamada, çoğu pratik süreç hem plan odaklı hem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



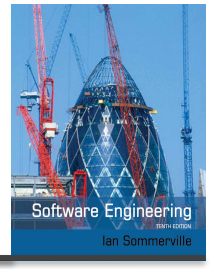
Yazılım süreç modelleri

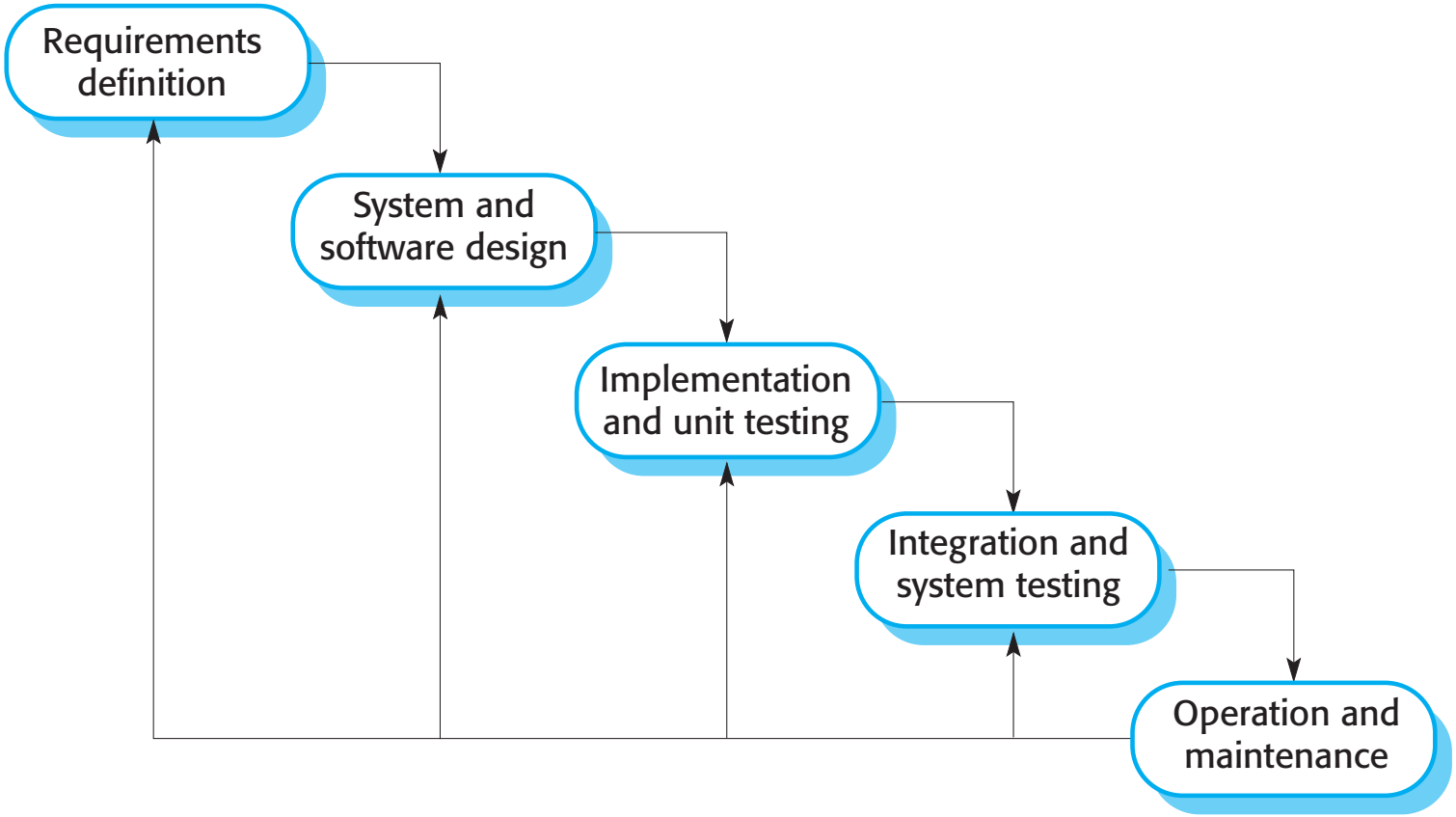
Yazılım süreç modelleri



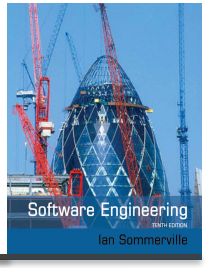
- ✧ şelale modeli
  - Plan odaklı model. Spesifikasyon ve geliştirmenin ayrı ve farklı aşamaları.
- ✧ artımlı geliştirme
  - Spesifikasyon, geliştirme ve doğrulama iç içe geçmiştir. Plan odaklı veya çevik olabilir.
- ✧ Entegrasyon ve yapılandırma
  - Sistem, mevcut yapılandırılabilir bileşenlerden birleştirilir. Plan odaklı veya çevik olabilir.
- ✧ Uygulamada, çoğu büyük sistem, tüm bu modellerin öğelerini içeren bir süreç kullanılarak geliştirilir.

Şelale (waterfall) modeli



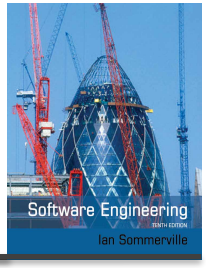


Şelale modeli aşamaları



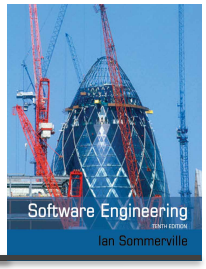
- Şelale modelinde ayrı tanımlanmış aşamalar vardır:
  - Gereksinim analizi ve tanımı
  - Sistem ve yazılım tasarımı
  - Gerçekleştirim ve birim testi
  - Entegrasyon ve sistem testi
  - Operasyon ve bakım
- Şelale modelinin en büyük dezavantajı, süreç başladıktan sonra değişime uyum sağlamanın zorluğudur. Prensip olarak, bir sonraki aşamaya geçmeden önce bir aşamanın tamamlanması gerekir.

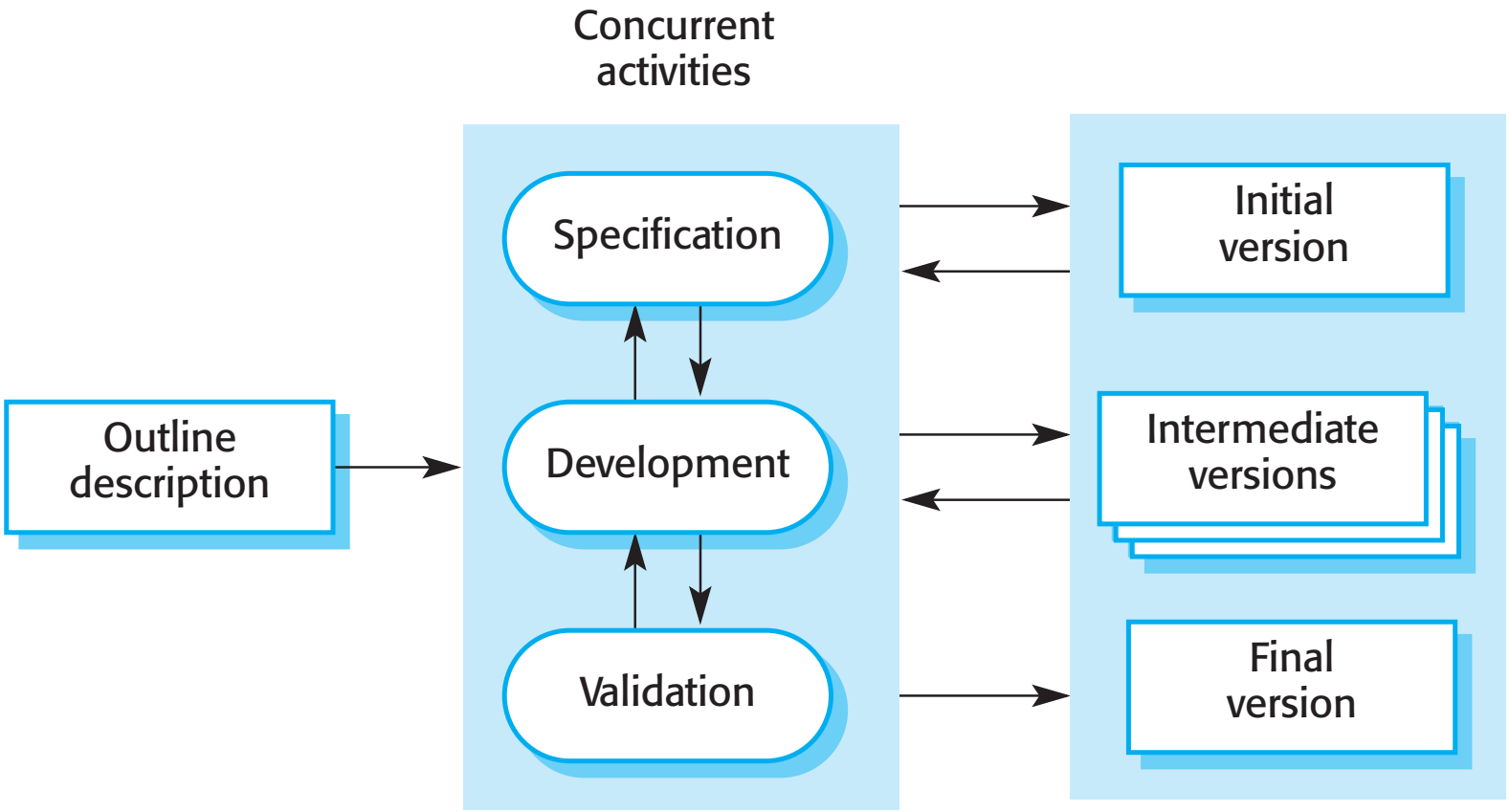
Şelale modeli problemleri



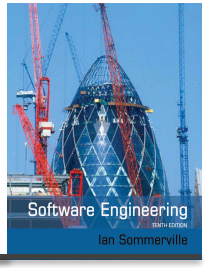
- Projenin esnek olmayan farklı aşamalara bölünmesi, değişen müşteri gereksinimlerine yanıt vermeyi zorlaştırıyor.
  - Bu nedenle, bu model yalnızca gereksinimler iyi anlaşıldığında uygundur ve tasarım sürecinde değişiklikler oldukça sınırlı olacaktır.
  - Birkaç iş sisteminin kararlı gereksinimleri vardır.
- Şelale modeli çoğunlukla, bir sistemin birkaç yerde geliştirildiği büyük sistem mühendisliği projeleri için kullanılır.
  - Bu durumlarda, şelale modelinin plana dayalı doğası, işin koordine edilmesine yardımcı olur.

Artımlı (Incremental) Geliştirme



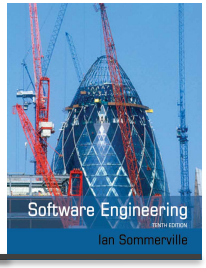


Artımlı geliştirme faydaları



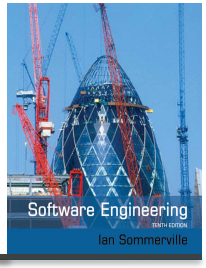
- Değişen müşteri gereksinimlerini karşılamanın maliyeti azalır.
  - Yeniden yapılması gereken analiz ve dokümantasyon miktarı, şelale modelinde gerekenden çok daha azdır.
- Yapılan geliştirme çalışmaları hakkında müşteri geri bildirimi almak daha kolaydır.
  - Müşteriler, yazılımın gösterimleri hakkında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

Artımlı geliştirme sorunları



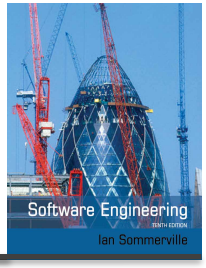
- Süreç görünmüyor.
  - Yöneticilerin ilerlemeyi ölçmek için düzenli çıktılara ihtiyacı vardır. Sistemler hızlı bir şekilde geliştirilirse, sistemin her versiyonunu yansıtan dokümanlar üretmek maliyet etkin değildir.
- Yeni artışlar eklendikçe sistem yapısı bozulma eğilimindedir.
  - Yazılımı geliştirmek için yeniden düzenleme için zaman ve para harcanmadıkça, düzenli değişiklik yapısını bozma eğilimindedir. Daha fazla yazılım değişikliğini dahil etmek giderek daha zor ve maliyetli hale geliyor.

Entegrasyon ve yapılandırma



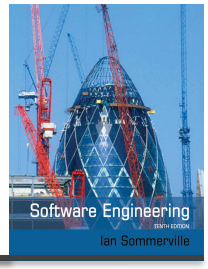
- Sistemlerin mevcut bileşenlerden veya uygulama sistemlerinden (bazen COTS -Ticari kullanıma hazır sistemler olarak adlandırılır) entegre edildiği yazılımın yeniden kullanımına dayalıdır.
- Yeniden kullanılan öğeler, davranışlarını ve işlevlerini kullanıcının gereksinimlerine uyarlamak için yapılandırılabilir
- Yeniden kullanım, artık birçok türde iş sistemi oluşturmak için standart yaklaşımdır.

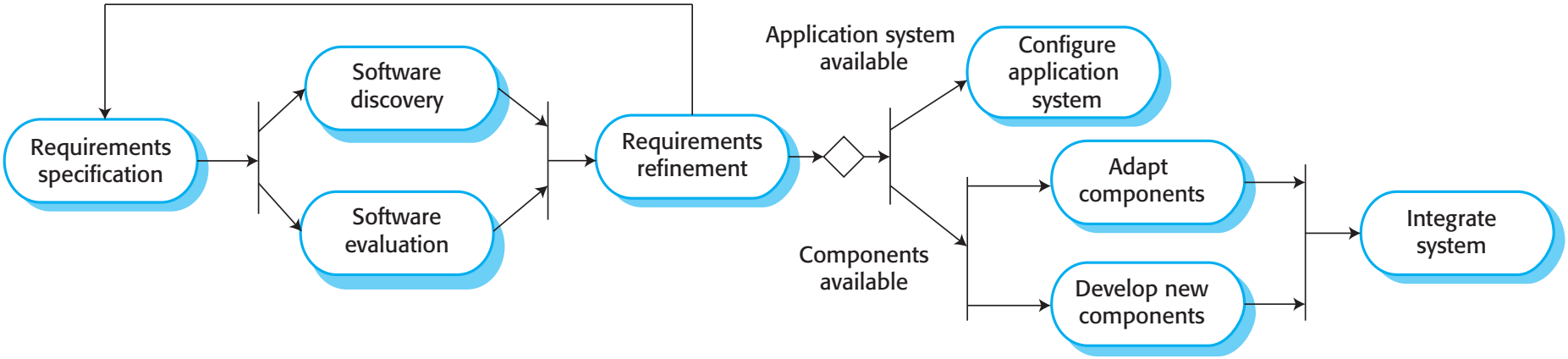
Yeniden kullanılabilir yazılım türleri



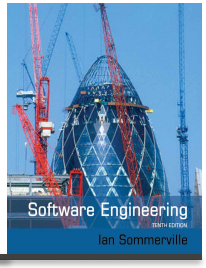
- Belirli bir ortamda kullanım için yapılandırılmış bağımsız uygulama sistemleri (bazen COTS olarak adlandırılır).
- .NET veya J2EE gibi bir bileşen çerçevesiyle bütünleştirilecek bir paket olarak geliştirilen nesneler.
- Servis standartlarına göre geliştirilen ve uzaktan çağrıya açık web hizmetleri.

Yeniden kullanım odaklı yazılım mühendisliği



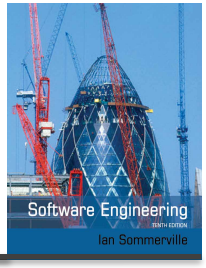


Anahtar süreç aşamaları

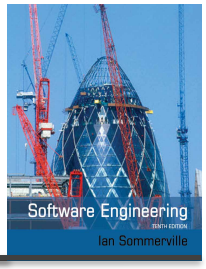


- ✧ Gereksinim belirtimi
- ✧ Yazılım keşfi ve değerlendirmesi
- ✧ Gereksinim iyileştirme
- ✧ Uygulama sistemi yapılandırması
- ✧ Bileşen uyarlaması ve entegrasyonu

Avantajlar ve dezavantajlar

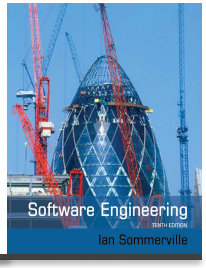


- Sıfırdan daha az yazılım geliştirildiğinden maliyetler ve riskler azalır
- Sistemin daha hızlı teslimatı ve dağıtımı
- Ancak gereksinimlerden ödün verilmesi kaçınılmazdır, bu nedenle sistem kullanıcıların gerçek ihtiyaçlarını karşılamayabilir.
- Yeniden kullanılan sistem öğelerinin gelişimi üzerinde kontrol kaybı



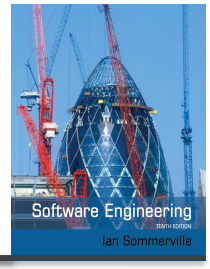
Süreç aktiviteleri

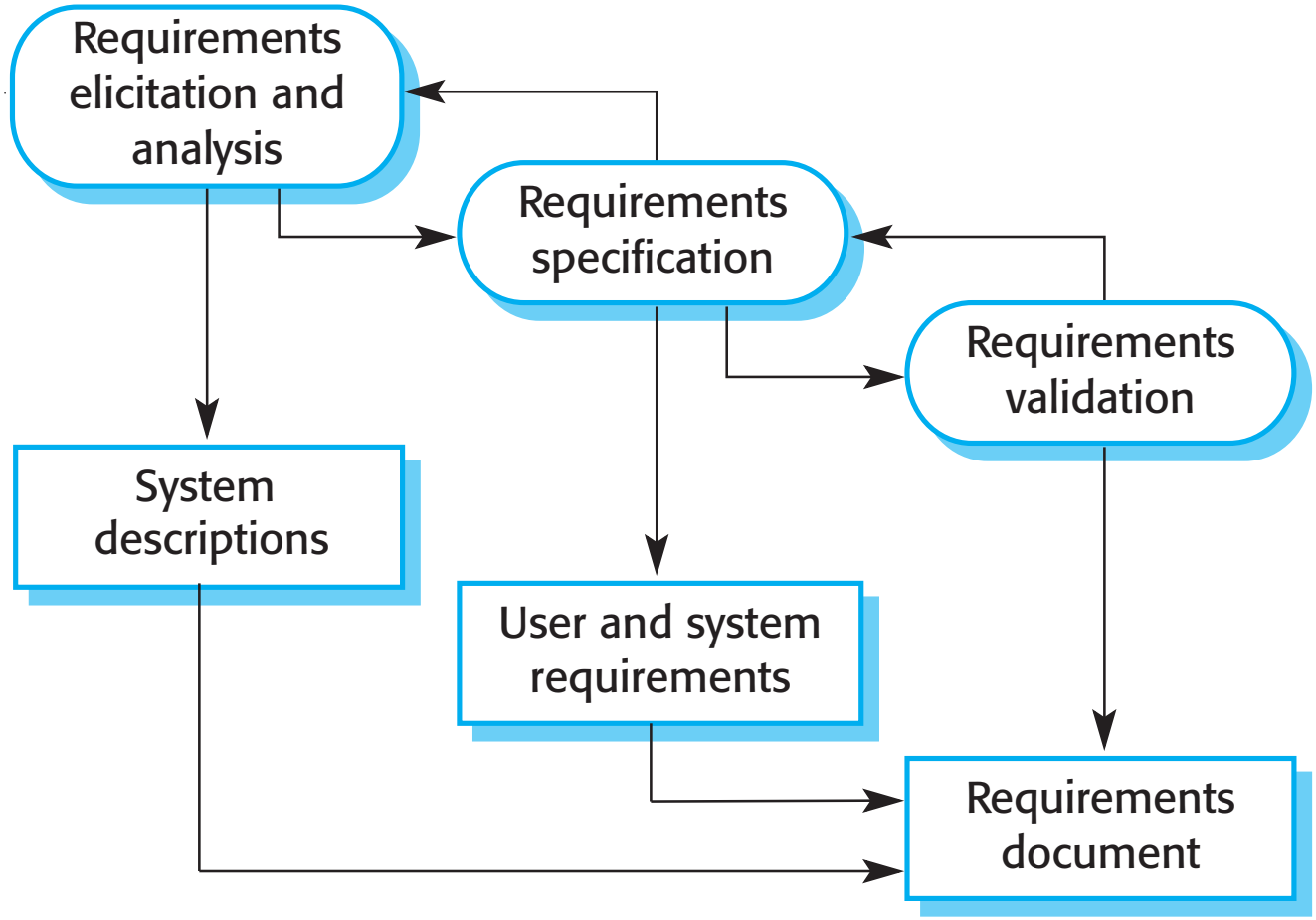
Süreç aktiviteleri



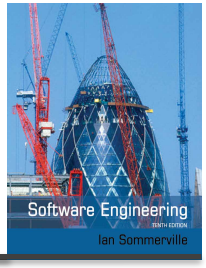
- Gerçek yazılım süreçleri, bir yazılım sistemini belirleme, tasarlama, uygulama ve test etme genel amacı ile teknik, işbirlikçi ve yönetimsel faaliyetlerin kırılımlı dizileridir.
- Spesifikasyon, geliştirme, doğrulama ve evrimden oluşan dört temel süreç etkinliği, farklı geliştirme süreçlerinde farklı şekilde organize edilmiştir.
- Örneğin, şelale modelinde sırayla düzenlenirler, oysa artımlı geliştirmede aralarına eklenirler.

Gereksinim mühendisliği süreci



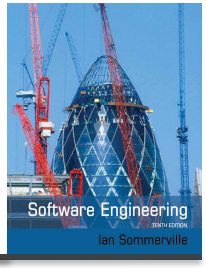


Yazılım spesifikasyonu



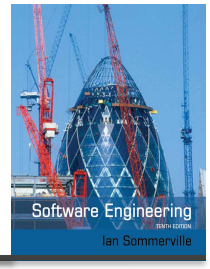
- Hangi hizmetlerin gerekli olduğunu ve sistemin çalışması ve geliştirilmesi üzerindeki kısıtlamaları belirleme süreci.
- Gereksinim mühendisliği süreci
  - Gereksinimlerin ortaya çıkarılması ve analizi
    - Sistem paydaşları sistemden ne talep ediyor veya bekliyor?
  - Gereksinim belirtimi
    - Gereksinimlerin ayrıntılı olarak tanımlanması
  - Gereksinim doğrulama
    - Gereksinimlerin geçerliliğini kontrol etme

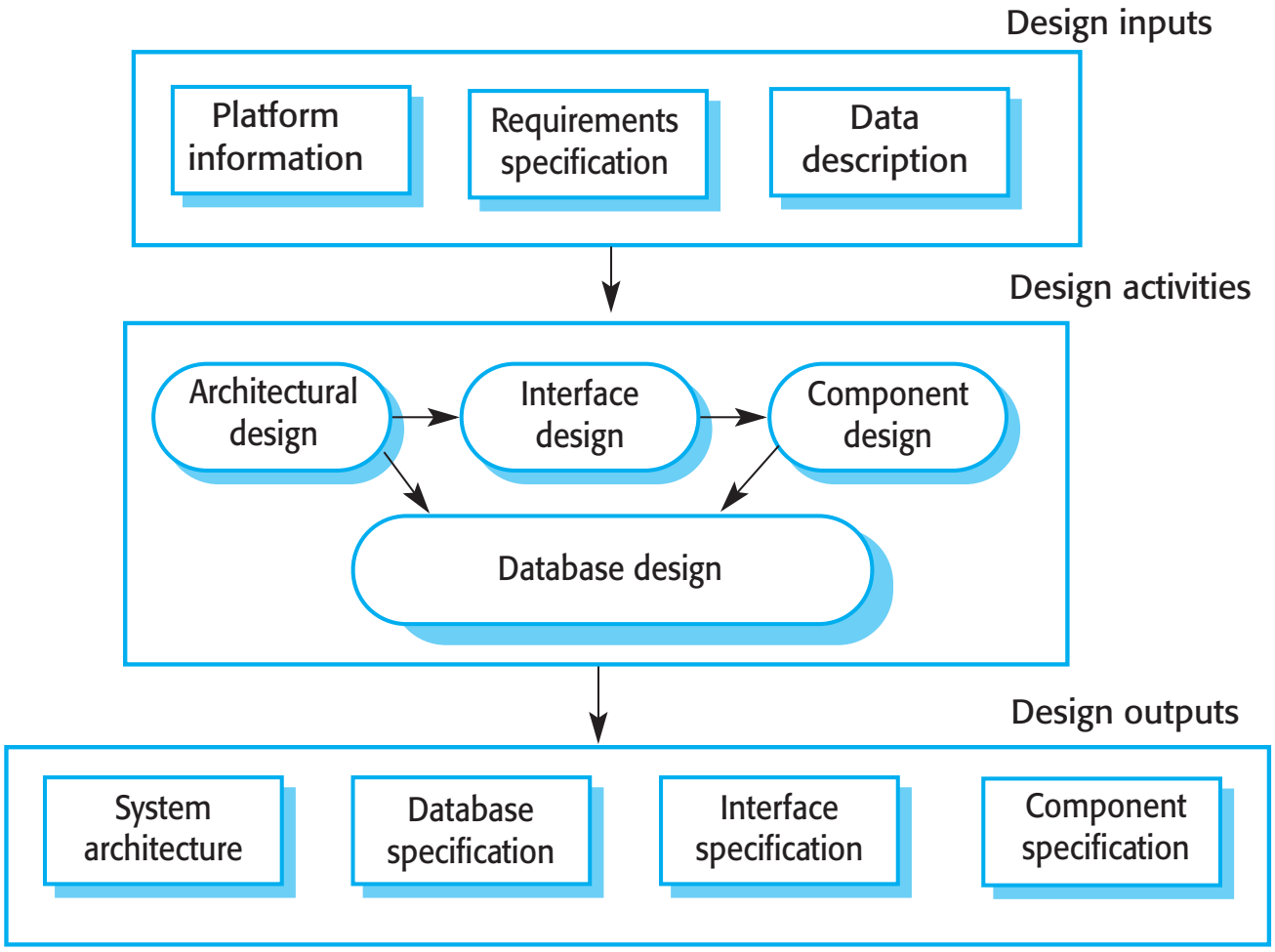
Yazılım tasarımı ve gerçekleştirilmesi



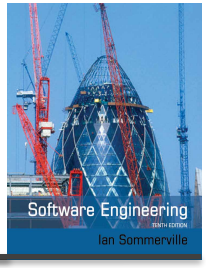
- Sistem belirtimini yürütülebilir bir sisteme dönüştürme işlemi.
- Yazılım Tasarımı
  - Spesifikasyonu gerçekleştiren bir yazılım yapısı tasarlayın;
- Gerçekleştirim
  - Bu yapıyı yürütülebilir bir programa çevirin;
- Tasarım ve gerçekleştirim faaliyetleri yakından ilişkilidir ve iç içe geçmiş olabilir.

Tasarım sürecinin genel bir modeli



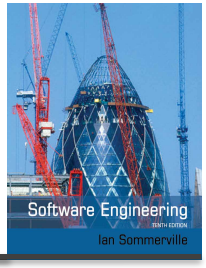


Tasarım Aktiviteleri



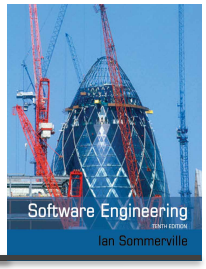
- *Sistemin genel yapısını, temel bileşenleri (alt sistemler veya modüller), bunların ilişkilerini ve bunların nasıl dağıtıldığını belirlediğiniz **mimari tasarım**.*
- *Sistem veri yapılarını ve bunların bir veritabanında nasıl temsil edileceğini tasarladığınız **veritabanı tasarımı**.*
- *Sistem bileşenleri arasındaki arayüzleri tanımladığınız **arayüz tasarımı**.*
- *Yeniden kullanılabilir bileşenleri aradığınız **bileşen seçimi ve tasarımı**. Mevcut değilse, nasıl çalışacağını siz tasarlarsınız.*

Sistem gerçekleştirimi



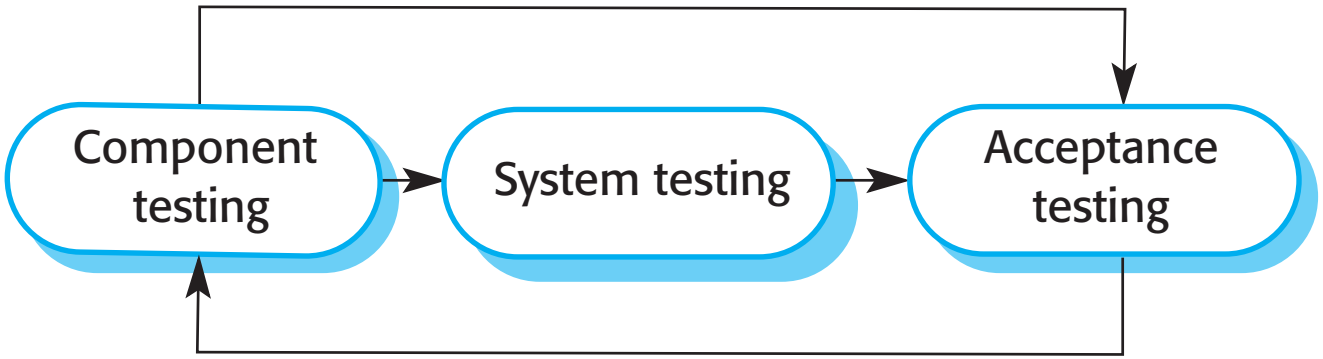
- Yazılım, bir program veya programlar geliştirerek veya bir uygulama sistemi yapılandırarak uygulanır.
- Tasarım ve uygulama, çoğu yazılım sistemi türü için serpiştirilmiş etkinliklerdir.
- Programlama, standart bir süreci olmayan bireysel bir faaliyettir.
- Hata ayıklama, program hatalarını bulma ve bu hataları düzeltme faaliyetidir.

Yazılım Doğrulama ve Geçerleme

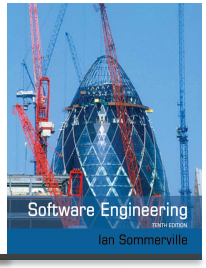


- Doğrulama ve geçerleme (V&V), bir sistemin spesifikasyonlarına uygun olduğunu ve sistem müşterisinin gereksinimlerini karşıladığını göstermeyi amaçlar.
- Kontrol ve gözden geçirme süreçleri ile sistem testini içerir.
- Sistem testi, sistem tarafından işlenecek gerçek verilerin spesifikasyonundan türetilen test senaryoları ile sistemin yürütülmesini içerir.
- Test etme, en sık kullanılan V&V etkinliğidir.

Test aşamaları

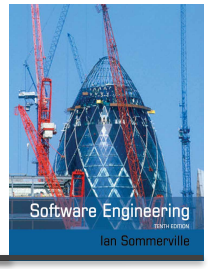


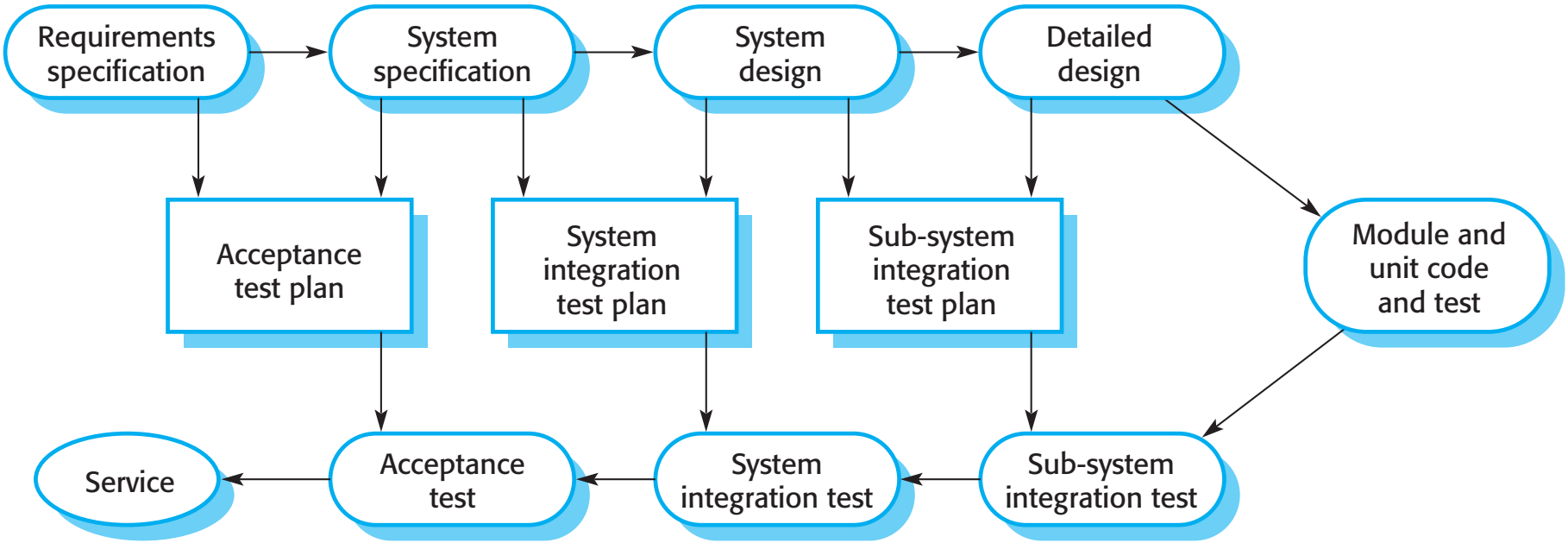
Test aşamaları



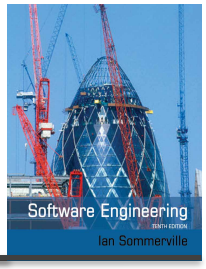
- Bileşen testi
  - Bireysel bileşenler bağımsız olarak test edilir;
  - Bileşenler, işlevler veya nesneler veya bu varlıkların tutarlı grupları olabilir.
- Sistem testi
  - Sistemin bir bütün olarak test edilmesi. Ortaya çıkan özelliklerin test edilmesi özellikle önemlidir.
- Müşteri testi
  - Sistemin müşterinin ihtiyaçlarını karşılayıp karşılamadığını kontrol etmek için müşteri verileriyle test etme.

Plan odaklı bir yazılım sürecindeki test aşamaları (V-modeli)



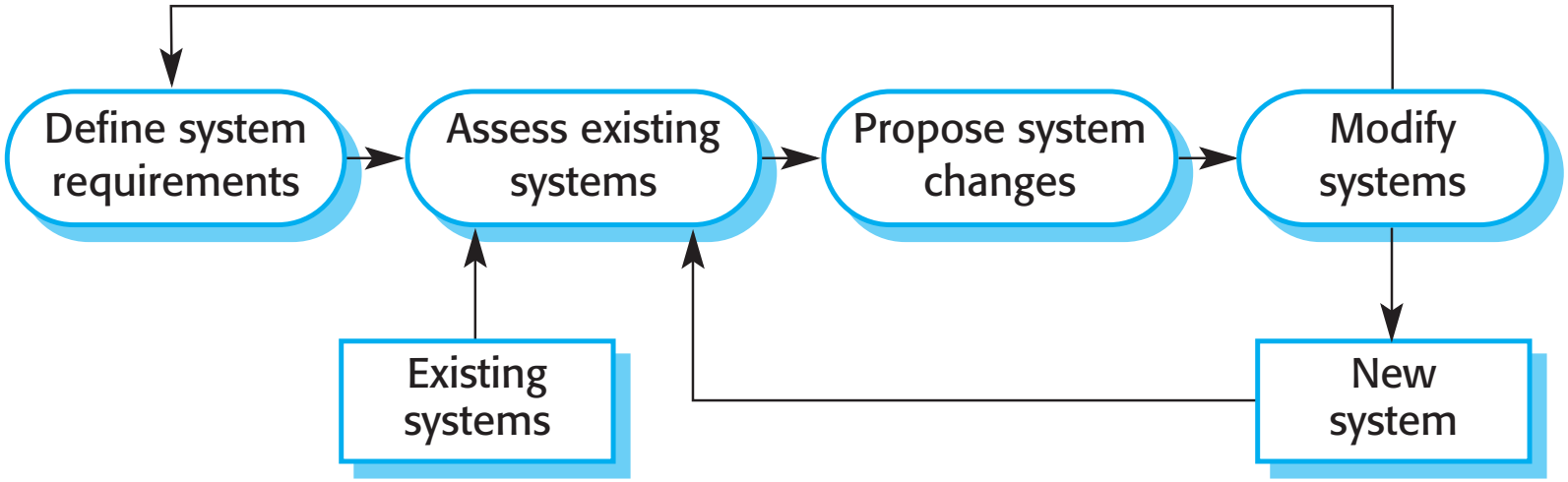


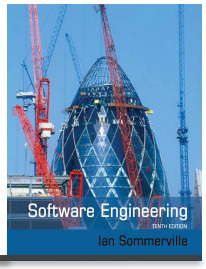
Yazılım evrimi



- Yazılım doğası gereği esnektir ve değişebilir.
- Değişen iş koşulları ile gereksinimler değiştikçe, işi destekleyen yazılımlar da gelişmeli ve değişmelidir.
- Geliştirme ve evrim (bakım) arasında bir sınır olmasına rağmen, giderek daha az sistem tamamen yeni olduğu için bu giderek önemsiz hale geliyor.

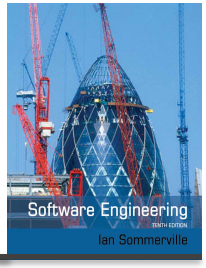
Sistem evrimi





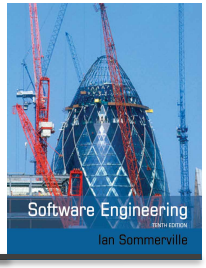
Değişimle başa çıkmak

Değişimle başa çıkmak



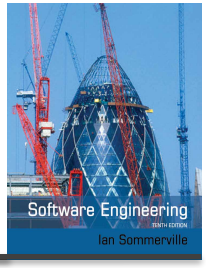
- Tüm büyük yazılım projelerinde değişim kaçınılmazdır.
  - İş değişiklikleri yeni ve değişen sistem gereksinimlerine yol açar
  - Yeni teknolojiler, uygulamaları iyileştirmek için yeni olanaklar açar
  - Değişen platformlar uygulama değişiklikleri gerektirir
- Değişiklik yeniden çalışmaya yol açar,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

Değişen gereksinimlerle başa çıkmak



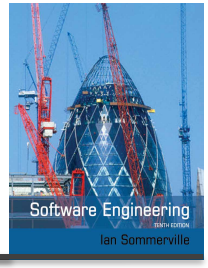
- Müşterinin gereksinimlerini ve tasarım kararlarının fizibilitesini kontrol etmek için sistemin bir versiyonunun veya sistemin bir bölümünün hızlı bir şekilde geliştirildiği **sistem prototipleme**. Bu yaklaşım, değişim beklentisini destekler.
- Sistem artımlarının yorum ve deneme için müşteriye teslim edildiği **artımlı teslimat (incremental delivery)**. Bu, hem değişiklikten kaçınmayı hem de değişiklik toleransını destekler.

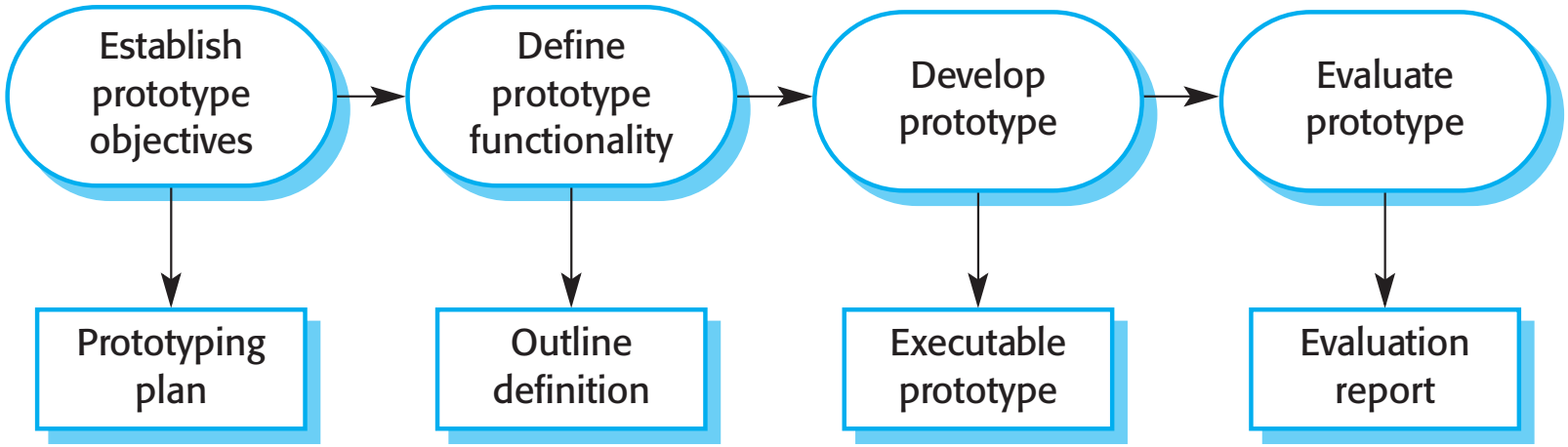
Yazılım prototipleme



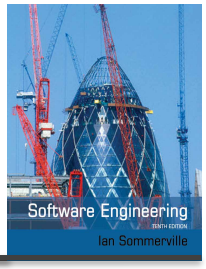
- Prototip, kavramları göstermek ve tasarım seçeneklerini denemek için kullanılan bir sistemin ilk versiyonudur.
- Bir prototip şu durumlarda kullanılabilir:
  - Gereksinimlerin ortaya çıkarılmasına ve doğrulanmasına yardımcı olmak için gereksinim mühendisliği süreci;
  - Seçenekleri keşfetmek ve bir UI tasarımı geliştirmek için tasarım süreçlerinde;
  - Test sürecinde arka arkaya testler yapmak için.

Prototip geliştirme süreci



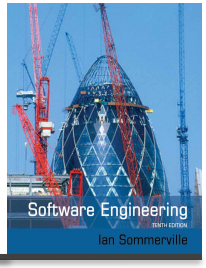


Kullanılıp atılan prototipler (Throw-away prototypes)



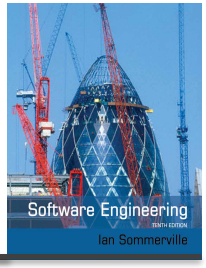
- Prototipler, bir üretim sistemi için iyi bir temel olmadığı için geliştirme sonrasında atılmalıdır:
  - Sistemi işlevsel olmayan gereksinimleri karşılayacak şekilde ayarlamak imkansız olabilir;
  - Prototipler normalde belgesizdir;
  - Prototip yapısı genellikle hızlı değişimle bozulur;
  - Prototip muhtemelen normal organizasyonel kalite standartlarını karşılamayacaktır.

artımlı teslimat (incremental delivery)



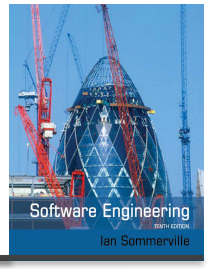
- Sistemi tek bir teslimat olarak teslim etmek yerine, geliştirme ve teslim, gerekli işlevselliğin bir kısmını sağlayan her bir artışla birlikte artımlara bölünür.
- Kullanıcı gereksinimlerine öncelik verilir ve en yüksek öncelikli gereksinimler erken artışlara dahil edilir.
- Bir parçanın geliştirilmesi başlatıldığında, gereksinimler dondurulur, ancak sonraki artışlar için gereksinimler gelişmeye devam edebilir.

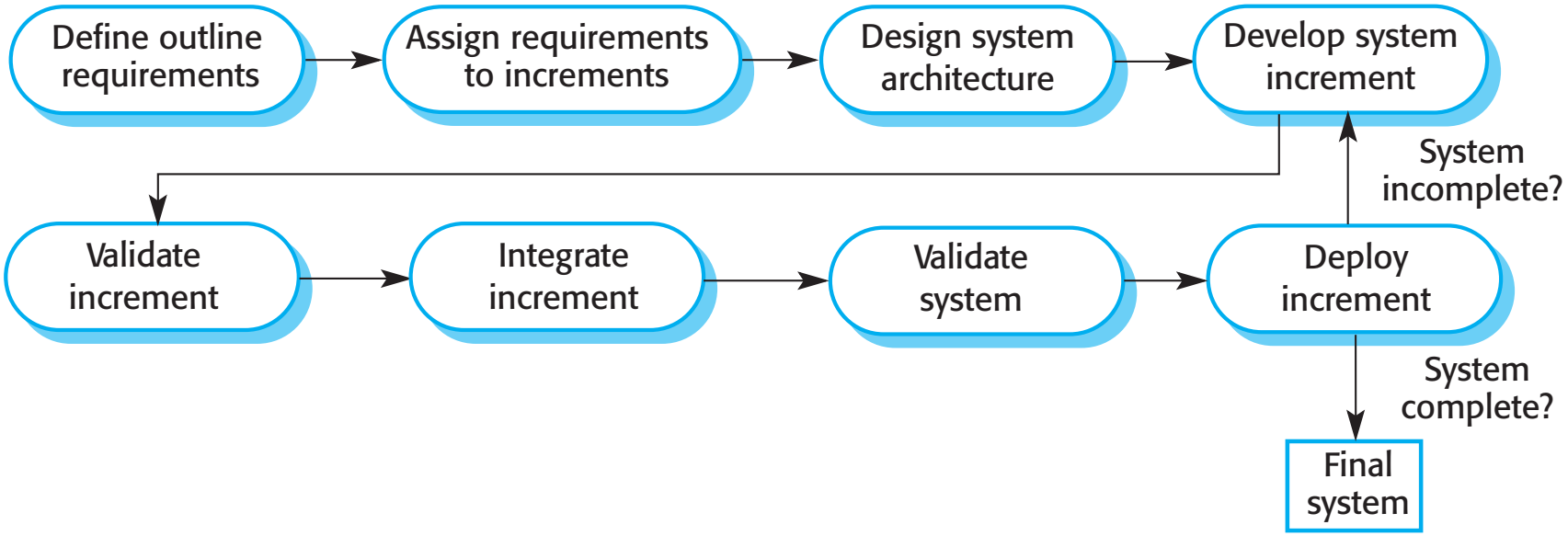
Artımlı geliştirme ve teslimat



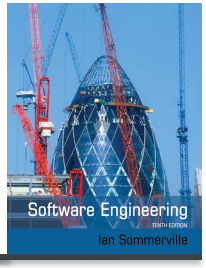
- artımlı geliştirme
  - Sistemi aşamalı olarak geliştirin ve bir sonraki parçanın geliştirilmesine geçmeden önce her bir parçayı değerlendirin;
  - Çevik yöntemlerde kullanılan normal yaklaşım;
  - Kullanıcı/müşteri vekili tarafından yapılan değerlendirme.
- artımlı teslimat
  - Son kullanıcılar tarafından kullanılmak üzere bir artış dağıtın;
  - Yazılımın pratik kullanımı hakkında daha gerçekçi değerlendirme;

artımlı teslimat (incremental delivery)

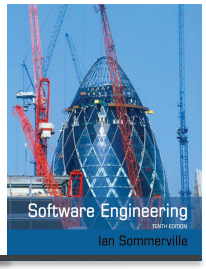




Artımlı teslimat avantajları

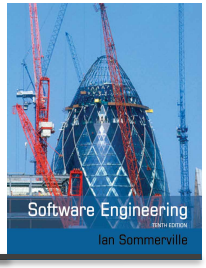


- Müşteri değeri, sistem işlevselliğinin daha erken kullanılabilir olması için her artımla birlikte teslim edilebilir.
- Erken artımlar, sonraki artımlar için gereksinimlerin ortaya çıkarılmasına yardımcı olmak için bir prototip görevi görür.
- Genel proje başarısızlığı riski daha düşüktür.
- En yüksek öncelikli sistem hizmetleri, en çok testi alma eğilimindedir.



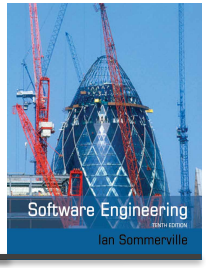
Süreç iyileştirme

Süreç iyileştirme



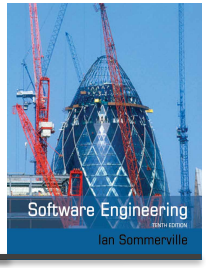
- Birçok yazılım şirketi, yazılımlarının kalitesini artırmanın, maliyetleri düşürmenin veya geliştirme süreçlerini hızlandırmanın bir yolu olarak yazılım süreci iyileştirmeye yönelmiştir.
- Süreç iyileştirme, ürün kalitesini artırmak ve/veya maliyetleri ve geliştirme süresini azaltmak için mevcut süreçleri anlamak ve bu süreçleri değiştirmek anlamına gelir.

Süreç iyileştirme faaliyetleri



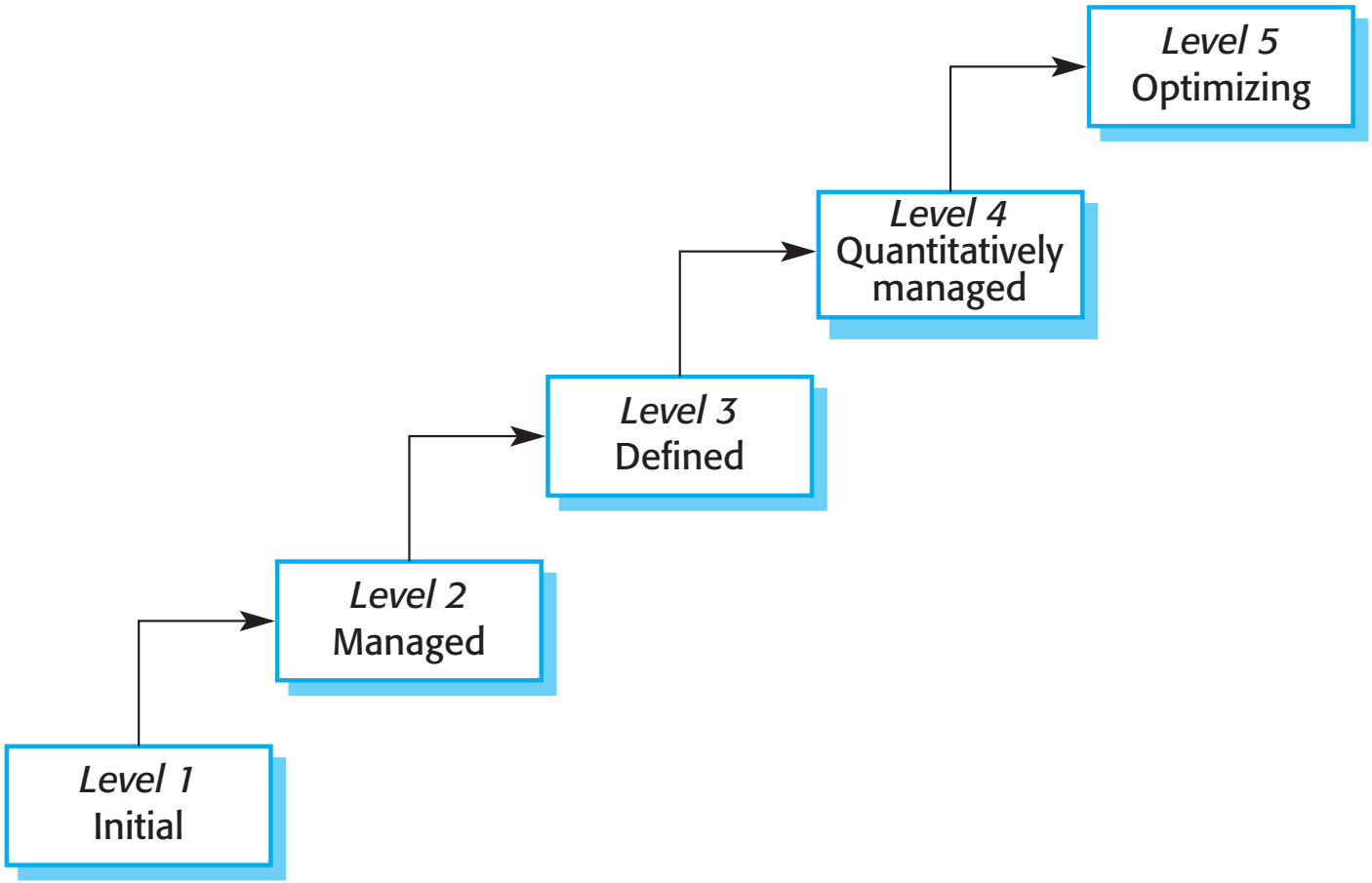
- *Süreç ölçümü*
  - *Yazılım sürecinin veya ürününün bir veya daha fazla özniteliğini ölçersiniz. Bu ölçümler, süreç iyileştirmelerinin etkili olup olmadığına karar vermenize yardımcı olan bir temel oluşturur.*
- *Süreç analizi*
  - *Mevcut süreç değerlendirilir ve süreç zayıflıkları ve darboğazlar belirlenir. Süreci tanımlayan süreç modelleri (bazen süreç haritaları olarak adlandırılır) geliştirilebilir.*
- *Süreç değişikliği*
  - *Belirlenen bazı süreç zayıflıklarını gidermek için süreç değişiklikleri önerilmiştir. Bunlar tanıtılır ve değişikliklerin etkinliği hakkında veri toplamak için döngü devam eder.*

Süreç metrikleri

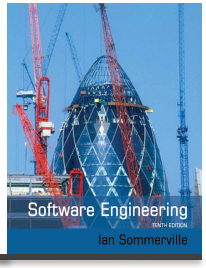


- Süreç aktivitelerinin tamamlanması için geçen süre
  - Örneğin. Bir etkinliği veya süreci tamamlamak için takvim zamanı veya işgücü.
- Süreçler veya faaliyetler için gerekli kaynaklar
  - Örneğin. Kişi-gün cinsinden toplam işgücü.
- Belirli bir olayın oluşum sayısı
  - Örneğin. Keşfedilen hataların sayısı.

Yetenek olgunluk seviyeleri



SEI yetenek olgunluk modeli



- ✧ Initial
  - Esasen kontrolsüz
- ✧ Repeatable
  - Tanımlanan ve kullanılan ürün yönetimi prosedürleri
- ✧ Defined
  - Tanımlanan ve kullanılan süreç yönetimi prosedürleri ve stratejileri
- ✧ Managed
  - Tanımlanan ve kullanılan kalite yönetim stratejileri
- ✧ Optimising
  - Tanımlanan ve kullanılan süreç iyileştirme stratejileri