Bölüm 2

Yazılım Süreçleri

Hedefler

- Bu bölümün amacı sizlere yazılım süreci kavramını tanıtmak - yazılım üretimi için gereken adımlar
- Bu bölümü işledikten sonra,
  - Yazılım süreçlerini ve yazılım süreç modellerini anlayacaksınız
  - Üç genel yazılım süreci modelini ve ne zaman kullanılmaları gerektiğini bileceksiniz
  - Gereksinim analizi, yazılım geliştirme, test ve evrim aşamalarında yapılan temel işleri bileceksiniz
  - Süreçlerin neden gereksinim ve tasarımdaki değişikliklerle başa çıkacak şekilde düzenlenmesi gerektiğini anlayacaksınız
  - Yeni koşullara uyum sağlayabilen yazılım süreçleri oluşturabilmek için Rasyonel Birleşik Süreç (Rational Unified Process)' in iyi yazılım mühendisliğine nasıl entegre olduğunu anlamak

Yazılım Süreci

- Temel varsayımlar
  - İyi bir yazılım istiyorsak süreci iyi yönetmeliyiz
  - İyi bir süreç yönetimi riskleri azaltır
- Risk yönetimi
  - Bir yazılım projesinde neler yanlış gidebilir?
  - Risk nasıl azaltılabilir?

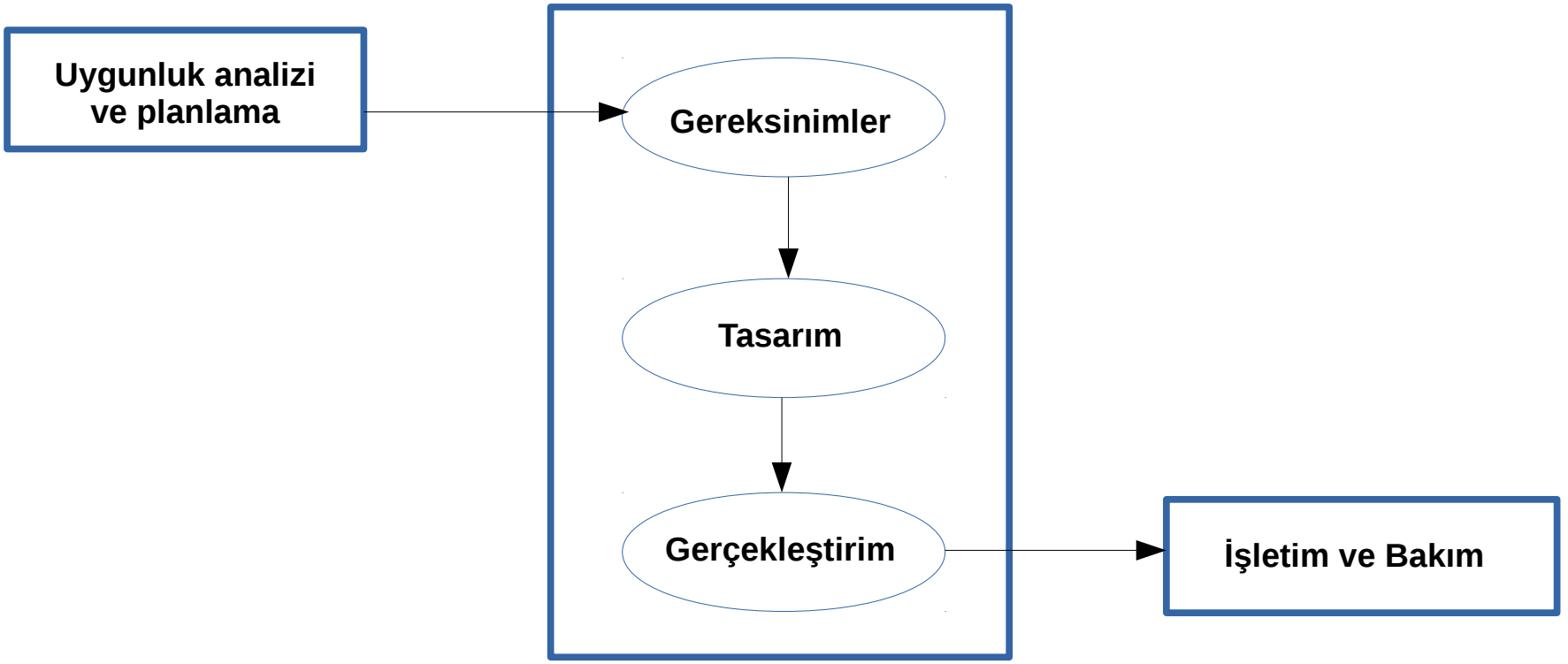
Yazılım Süreci

- Bir yazılım süreci, bir yazılım ürünü ortaya çıkarmak için yapılması gereken, birbiri ile ilişkili işlemler bütünüdür.
- Yazılım süreçleri farklılıklar gösterebilir ancak temel olarak dört işi içermelidir:
  - ***Yazılımı tanımlama (software specification)*** - Yazılımın ne yapacağı, kısıtları tanımlanmalıdır.
  - ***Yazılım tasarımı ve gerçekleştirim (software design and implementation)*** – Yazılım, tanımına uygun olarak üretilmelidir.
  - ***Yazılımın doğrulanması (software validation)*** – Yazılımın müşterinin isteklerini karşılayabildiği doğrulanmalıdır.
  - ***Yazılımın evrimi (software evolution)*** – Yazılım müşterinin değişen ihtiyaçlarına cevap verecek şekilde değiştirilebilmelidir.

Yazılım Süreci

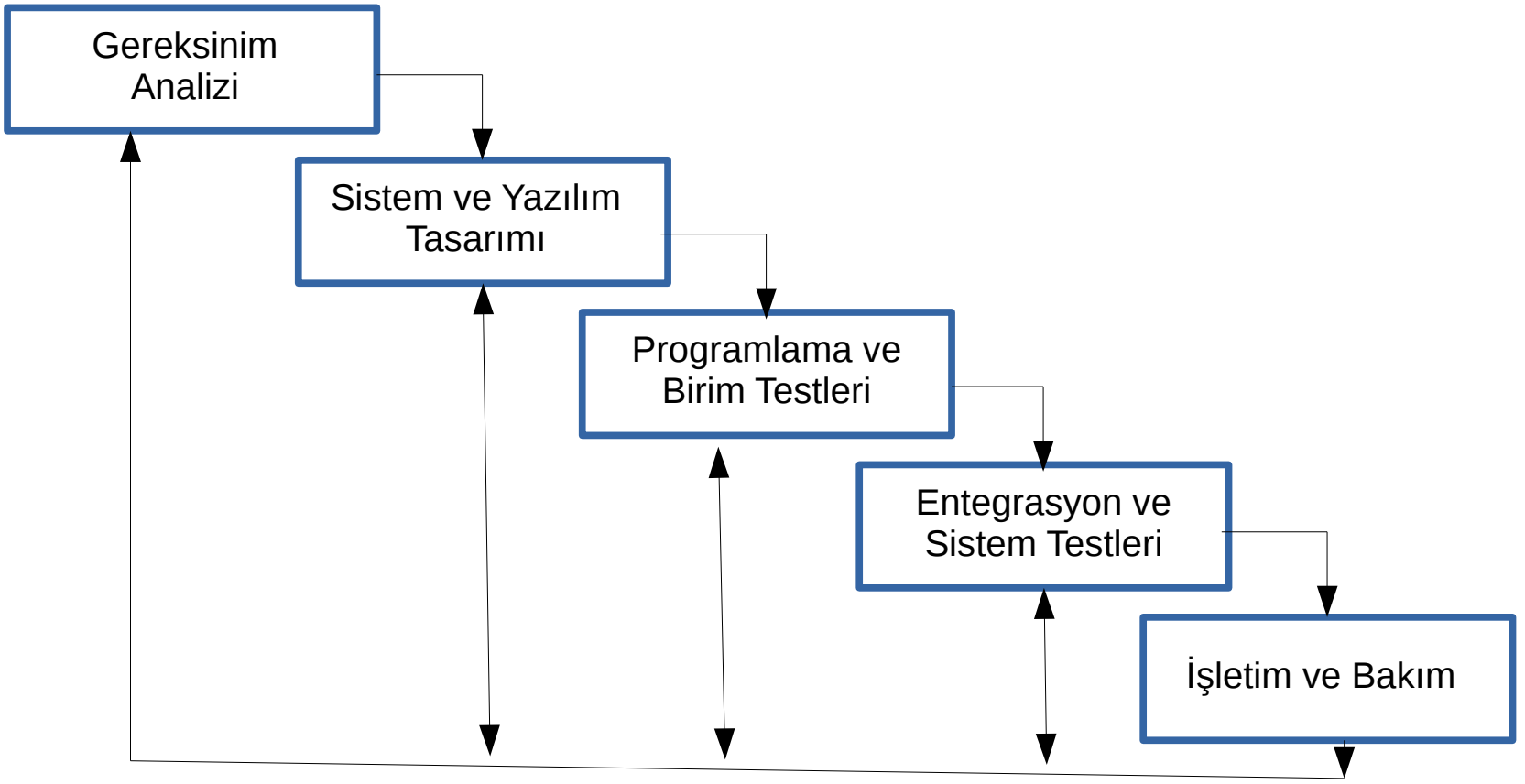
- Süreci sadece veri modeli belirleme, kullanıcı arayüzü tasarımı gibi aktiviteler içermez
- Süreç şunları da içerebilir:
  - Ürünler
  - Roller
  - Ön ve son koşullar

Yazılım Süreci (basitleştirilmiş)



Süreç Modelleri

- ***Şelale (Waterfall) Modeli***



Gereksinim Analizi

- Uygunluk çalışması
- Gereksinim analizi
- Gereksinim tanımları
- Gereksinim teknik şartnamesi

Sistem ve Yazılım Tasarımı

- Sistem tasarımı – Genel sistem mimarisi, kullanılacak yazılım, donanım, eğitim, ...
- Yazılım tasarımı – Yazılım bileşenlerinin tasarlanması
  - Unified Modelling Language (UML)

Programlama ve Birim Testleri

- Yazılım, programlar veya program birimleri bütünü olarak algılanmaktadır
- Birim testleri, her birimin doğru ve tanımına uygun çalıştığını onaylamak içindir

Entegrasyon ve Sistem Testleri

- Bağımsız program birimlerinin birlikte çalışabilirliğinin kontrol edilmesi
- Gereksinimlerin karşılandığının onaylanması
- Sistemin müşteriye teslim edilmesi

İşletim ve Bakım

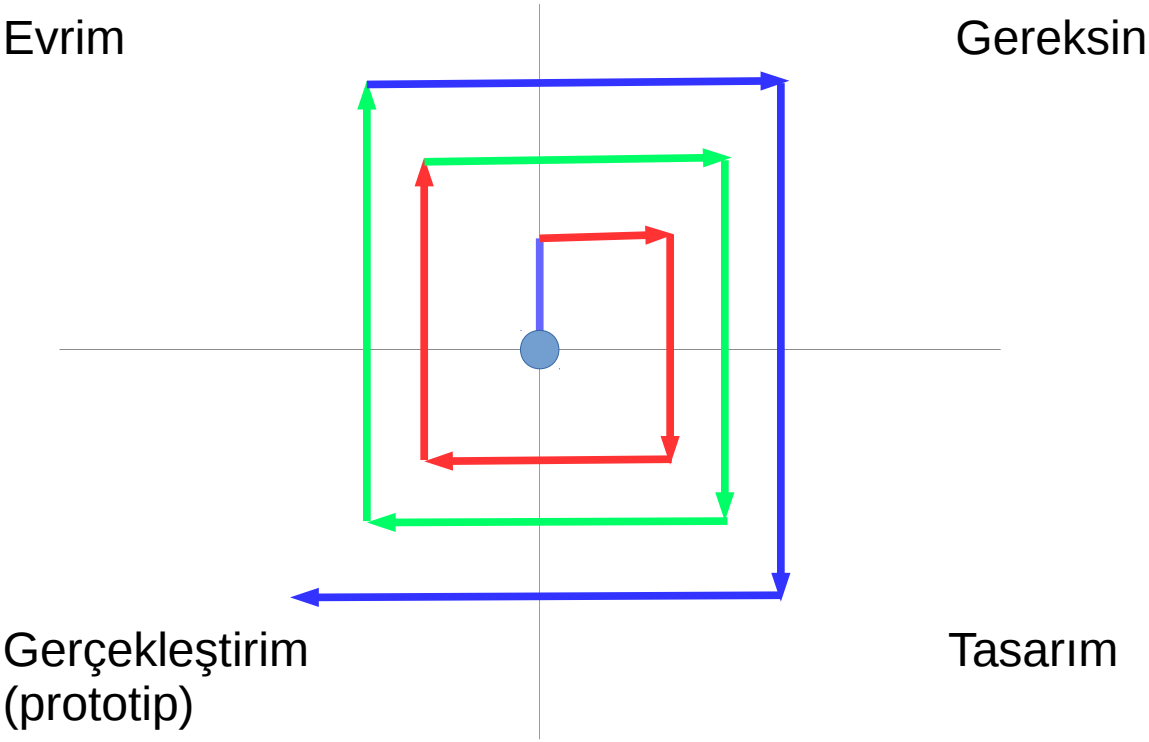
- Yazılımın kullanılmaya başlanması
- Hataların ve problemlerin belirlenip düzeltilmesi
- Zaman içinde yazılımın yeni gereksinimlere veya teknolojilere göre güncellenmesi; yazılıma eklentiler yapılması
- Yazılımla ilgili sürecin bitişi

Şelale (Waterfall) Modeli

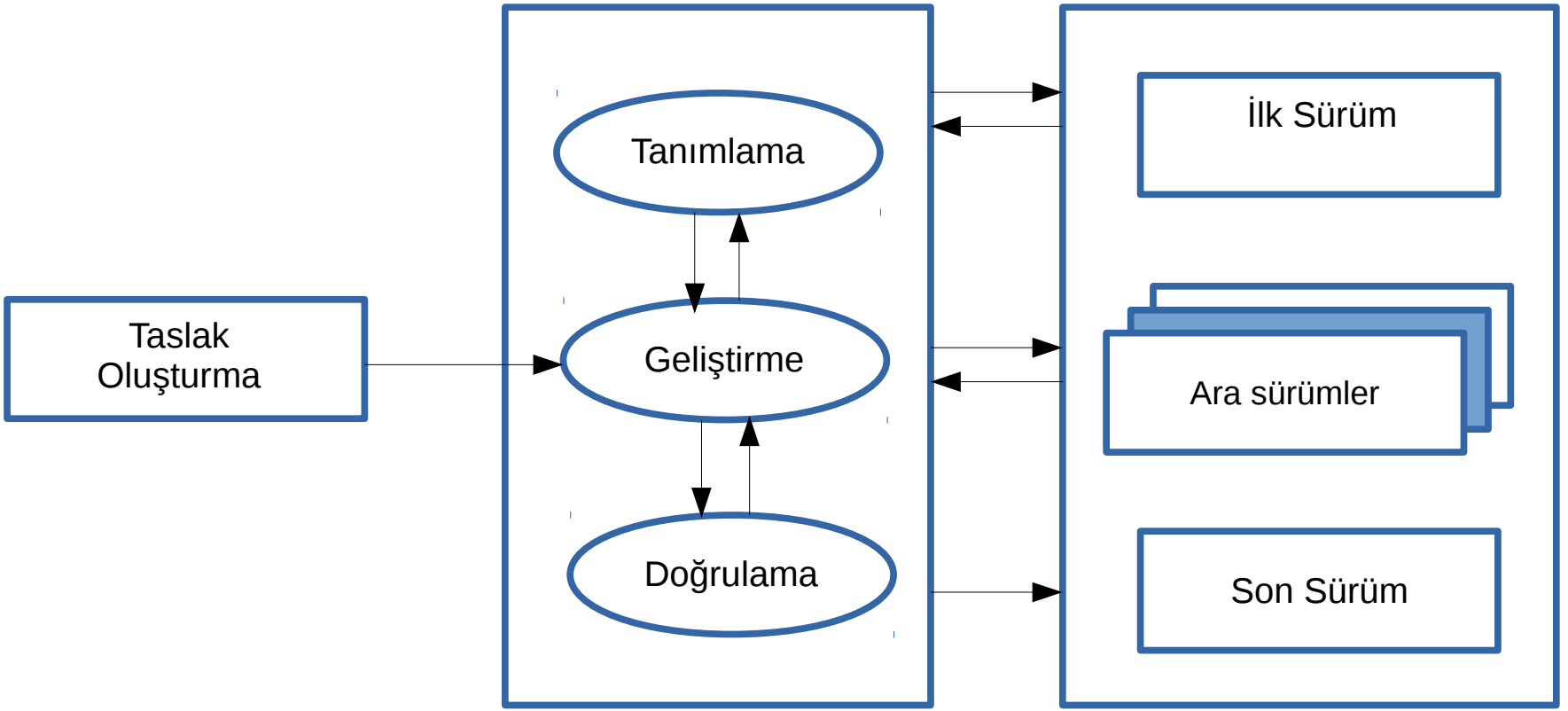
- Avantajları
  - Süreç görünürlüğü var– Sürecin hangi aşamasındayız, bitimine ne kadar kaldı...
  - Sorumluyu bulmak kolay
  - Kalite kontrol
  - Maliyet kontrolü
- Dezavantajları
  - Bir aşama bitmeden diğer aşamaya geçme konusunda esnek olmayışı

Süreç Modelleri

- ***Artımlı (Incremental) Geliştirim Modeli***



Artımlı (Incremental) Geliştirim Modeli



Artımlı (Incremental) Geliştirim Modeli

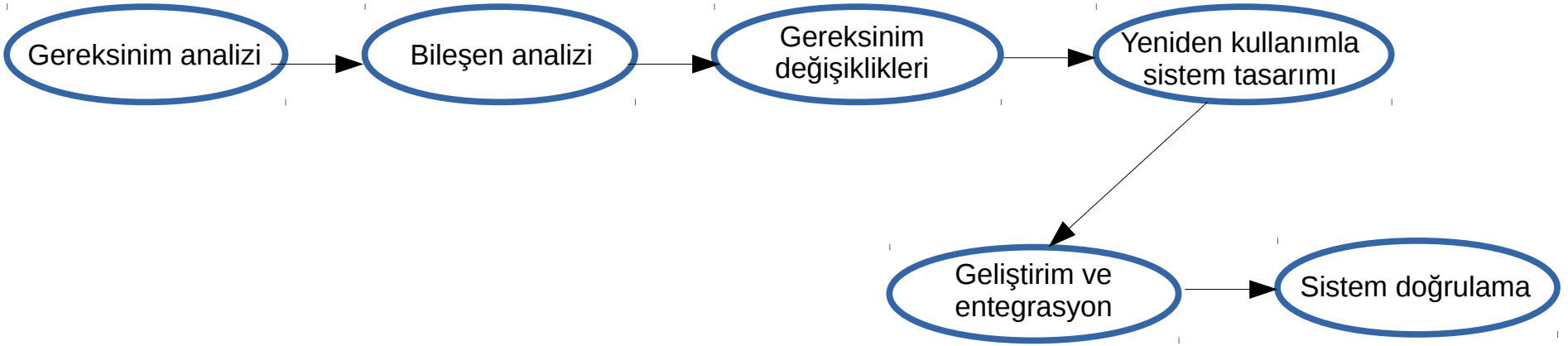
- Çevik (agile) yazılım geliştirme yaklaşımlarının ana kısmını oluşturur
- Çoğu iş, e-ticaret ve kişisel yazılımlar için Şelale'den daha iyi bir modeldir
  - Değişen müşteri isteklerini karşılamak daha az maliyetlidir
  - Müşteriden geri bildirim almak daha kolaydır
  - Yazılımın tamamı olmasa da müşteriye teslimat daha hızlıdır
  - Yeniden kullanılabilir modüller üretilebilir

Artımlı (Incremental) Geliştirim Modeli

- Yönetim bakış açısı ile iki problemi vardır
  - Süreç görünür değil (projenin neresindeyiz)
  - Refactoring yapılmazsa, yeni adımlar yazılımın yapısını bozabilir
- Bu problemler büyük, karmaşık ve uzun ömürlü sistemler için büyüktür

Süreç Modelleri

- ***Yeniden Kullanım Tabanlı (Reuse-Oriented) Yazılım Mühendisliği***



Yeniden Kullanım Tabanlı Model

- İlk ve son adımlar diğer modellerdeki gibi ancak ara adımlar farklıdır
  - Bileşen analizi
  - Gereksinim değişiklikleri
  - Yeniden kullanımla sistem tasarımı
  - Geliştirim ve entegrasyon

Yeniden Kullanım Tabanlı Model

- Zaman
- Maliyet
- Risk
- Gerçek isteklerin karşılanması
- Yeniden kullanılan bölümlerin kontrolü

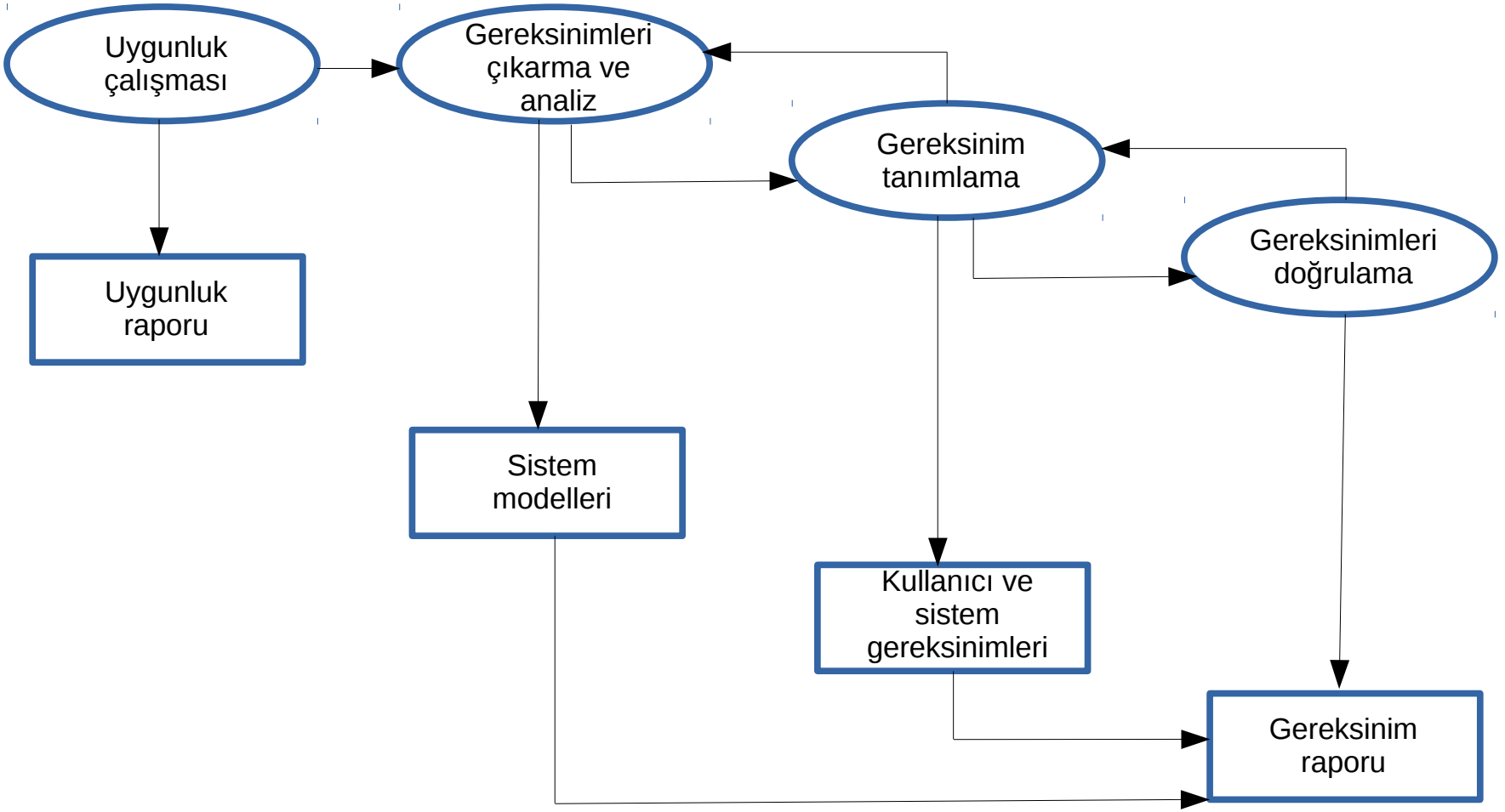
Süreçte Yapılan İşler

- Farklı modellerde yapılan işler farklılık gösterebilir (örn. Test)
- Yazılımın türü, kişiler ve organizasyon yapısı da işlerin farklı yürütülmesinin nedenleridir
- Hangi işler?
  - Yazılımı tanımlama
  - Yazılımın tasarımı ve gerçekleştirimi
  - Yazılımın doğrulanması
  - Yazılımın evrimi

Yazılımı Tanımlama

- Gereksinim mühendisliği
  - Yazılım ne iş yapacak, kısıtlar neler olacak?
  - Sürecin başında yapılan hatalar daha sonraki adımları de etkiler
- Bu süreçte dört temel iş bulunur:
  - Uygunluk çalışması
  - Gereksinimleri çıkarma ve analiz
  - Gereksinim tanımlama (doküman)
  - Gereksinimleri doğrulama

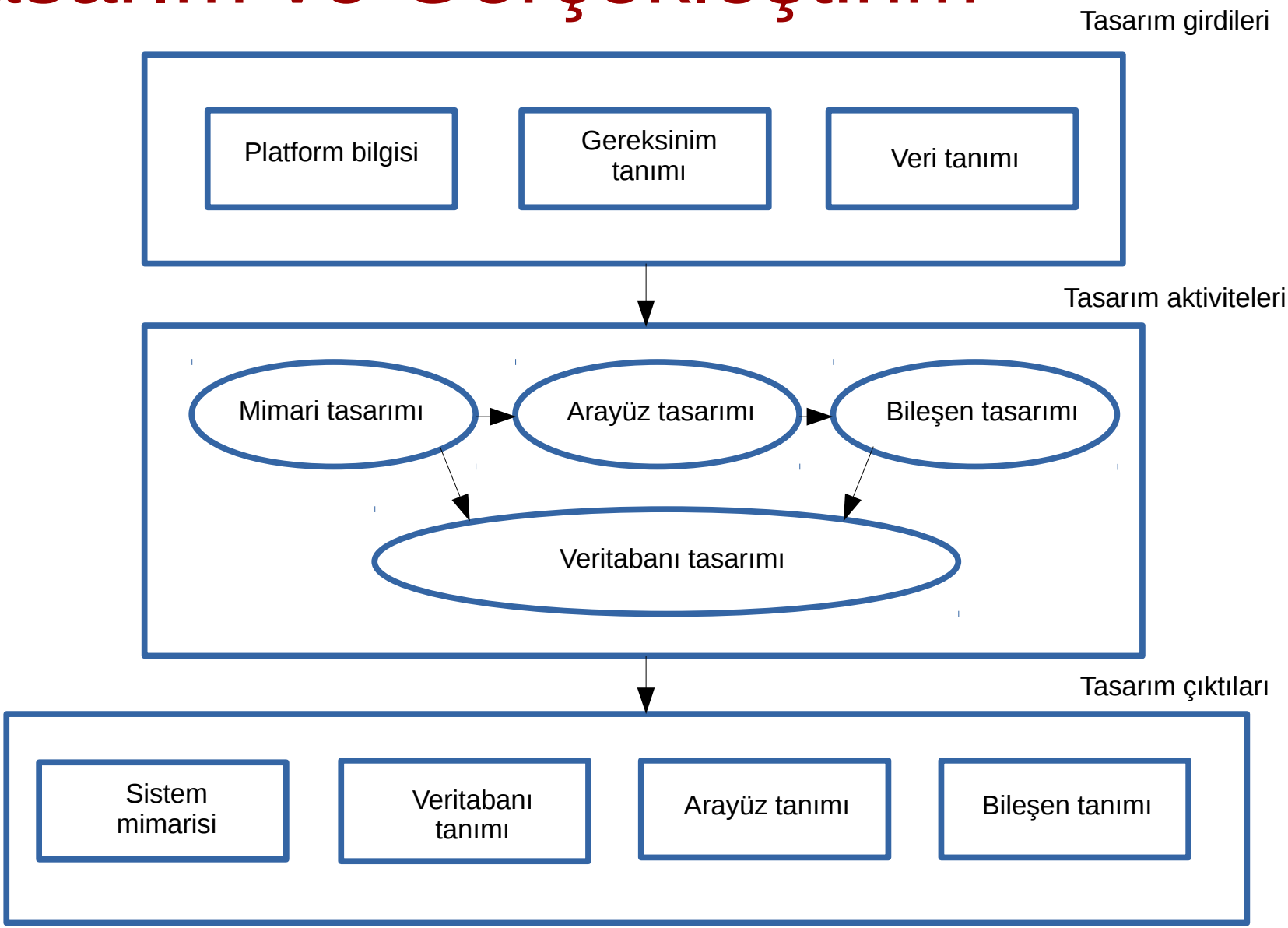
Yazılımı Tanımlama



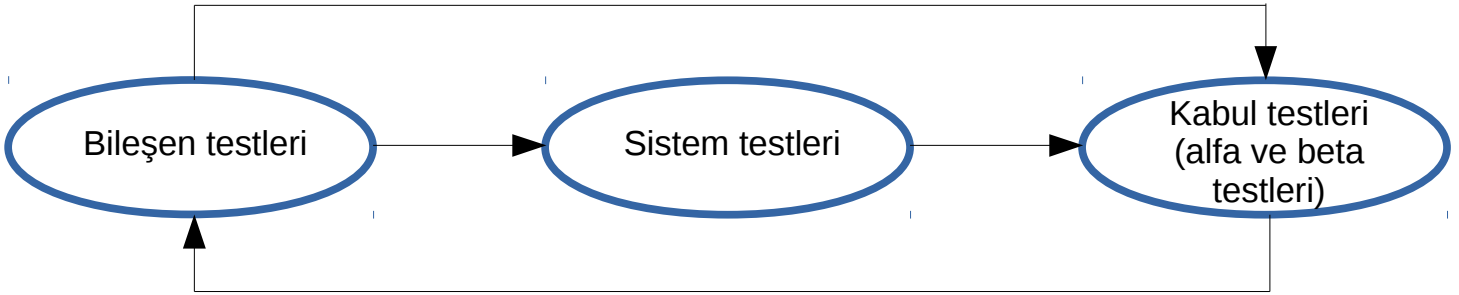
Tasarım ve Gerçekleştirim

- Aktiviteler sıralı veya dönüşümlü olabilir, adım tekrarı kaçınılmazdır
- Yazılım hangi ortamlarda çalışacak
- Bu süreçteki işler:
  - Mimari tasarım
  - Arayüz tasarımı
  - Bileşen tasarımı
  - Veritabanı tasarımı

Tasarım ve Gerçekleştirim



Yazılımın Doğrulanması

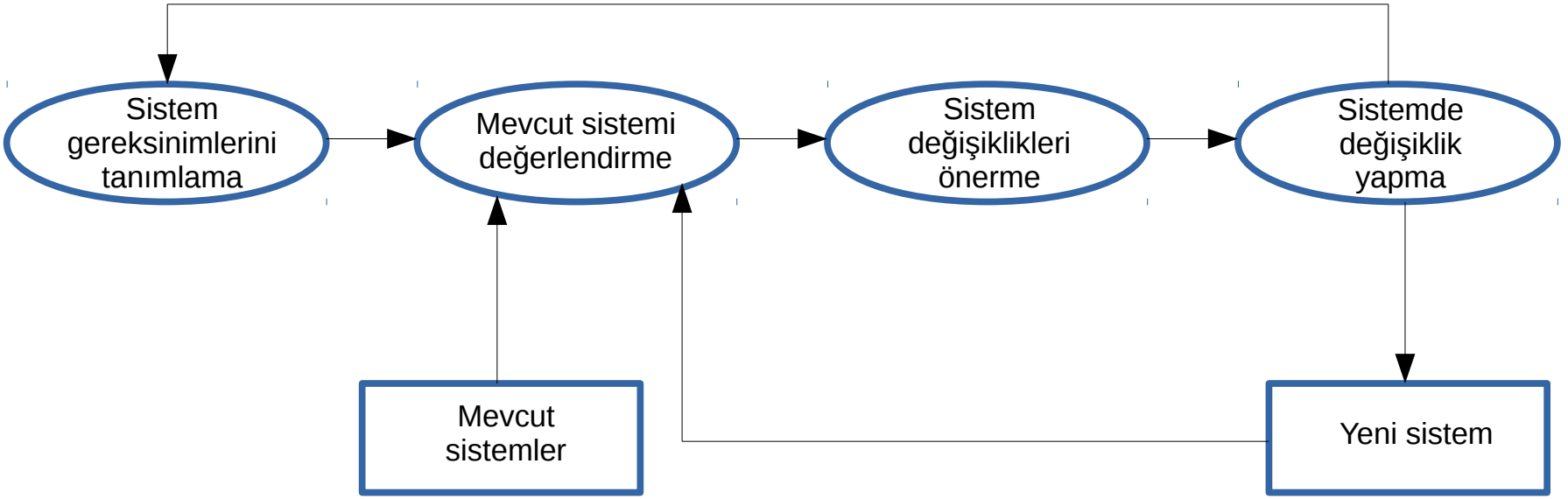


- Test verisi ile programı test etme
- Küçük sistemler hariç, tüm program tek bir bütün halinde test edilmemelidir
- Tekrarlı bir süreç
- Programcı kendi yazdığı bileşeni test edebilir
- Extreme proglamlamada testler gereksinimlerle birlikte yazılır
- Plan güdümlü yazılım

Yazılımın Evrimi

- Yazılım, değişen ve gelişen müşteri istekleri ve dünya şartlarına uyum sağlamalı
- Geliştirme ve bakım ayrı işler gibi düşünülmemeli
- Bakım geliştirmenin devamıdır

Yazılımın Evrimi



Değişikliklerle Baş Etme

- Büyük yazılım projelerinde değişim kaçınılmazdır
- Kullanılan modeli yazılımın değişimlerini destekleyebilmelidir
- Değişiklik maliyeti etkiler (rework)
  - Değişiklikten kaçınma (prototip üretme)
  - Değişikliğe tolerans gösterme (artımlı geliştirim)

Değişikliklerle Baş Etme

- Değişimle ve değişen sistem gereksinimleriyle baş etme
  - Sistem prototipi oluşturma – değişiklikten kaçınma
  - Artımlı geliştirim – hem değişiklikten kaçınma hem de değişikliğe kolayca uyum sağlama
  - Refactoring – değişiklikle baş etmek için önemli bir mekanizma (Bölüm 3)

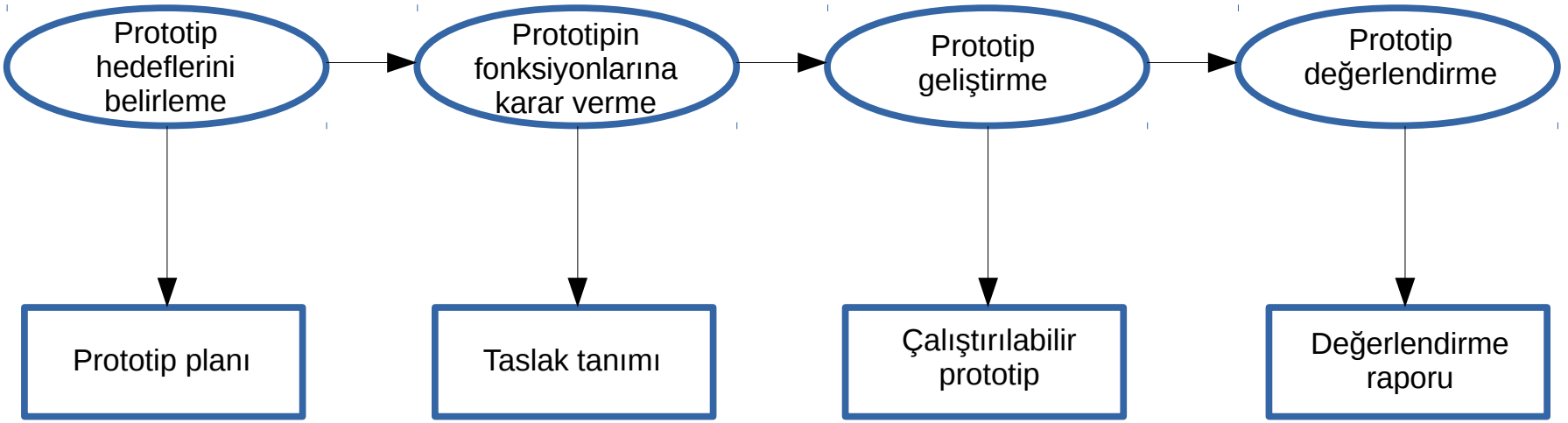
Prototip Oluşturma

- Prototip, bir yazılım sisteminin başlangıçtaki sürümüdür; programın yapabileceklerinin, tasarımının bir demosudur; problemleri ve olası çözümleri belirleyebilmek için kullanılır
- Prototipin hızlı ve artımlı geliştirimi, maliyet kontrolü ve müşterinin sürecin başlarında kullanım deneyimlerini aktarması açısından çok önemlidir
- A software prototype can be used in a software development process to help anticipate changes that may be required:

Prototip Oluşturma

- Prototipler sayesinde kullanıcı, sistemin ihtiyaçlarını ne ölçüde karşıladığını görebilir
  - Yeni fikirler
  - Sistemin zayıf ve güçlü yanları
- Prototip geliştikçe gereksinimlerdeki hatalar farkedilebilir ve bazı maddeler iptal edilebilir
- Prototip geliştirme hedefleri başlangıçta açıkça ortaya koyulmalıdır

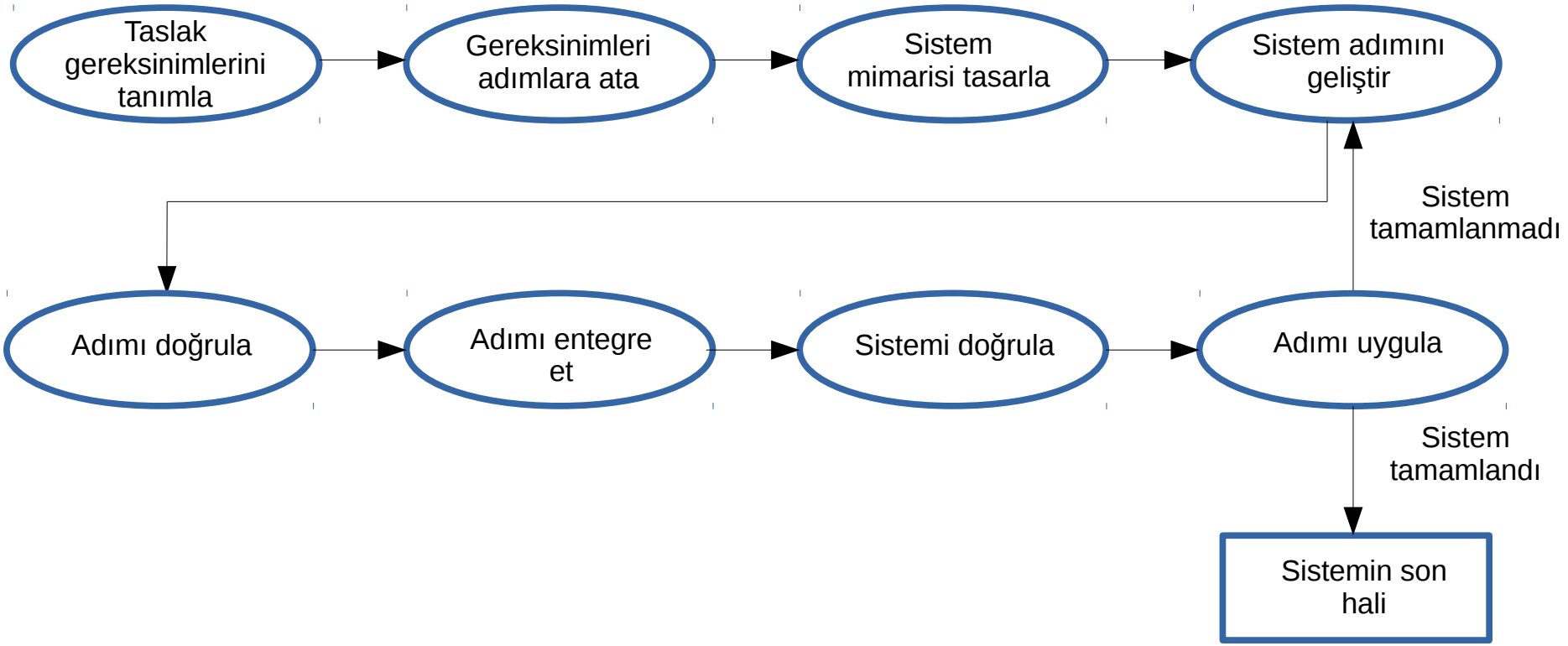
Prototip Oluşturma



Artımlı Teslimat (Incremental Delivery)

- Yazılımın bir kısmı müşteriye teslim edilir ve gerçek ortamda kullanılmaya başlanır
- Önce müşteriler sistemi taslak olarak tanımlar; hangi kısımlar daha önemli vb.
- Adım sayısı belirlenir, adımların fonksiyonellikleri tanımlanır
- İlk adım için gereksinim analizi yapılır

Artımlı Teslimat



Artımlı Teslimat

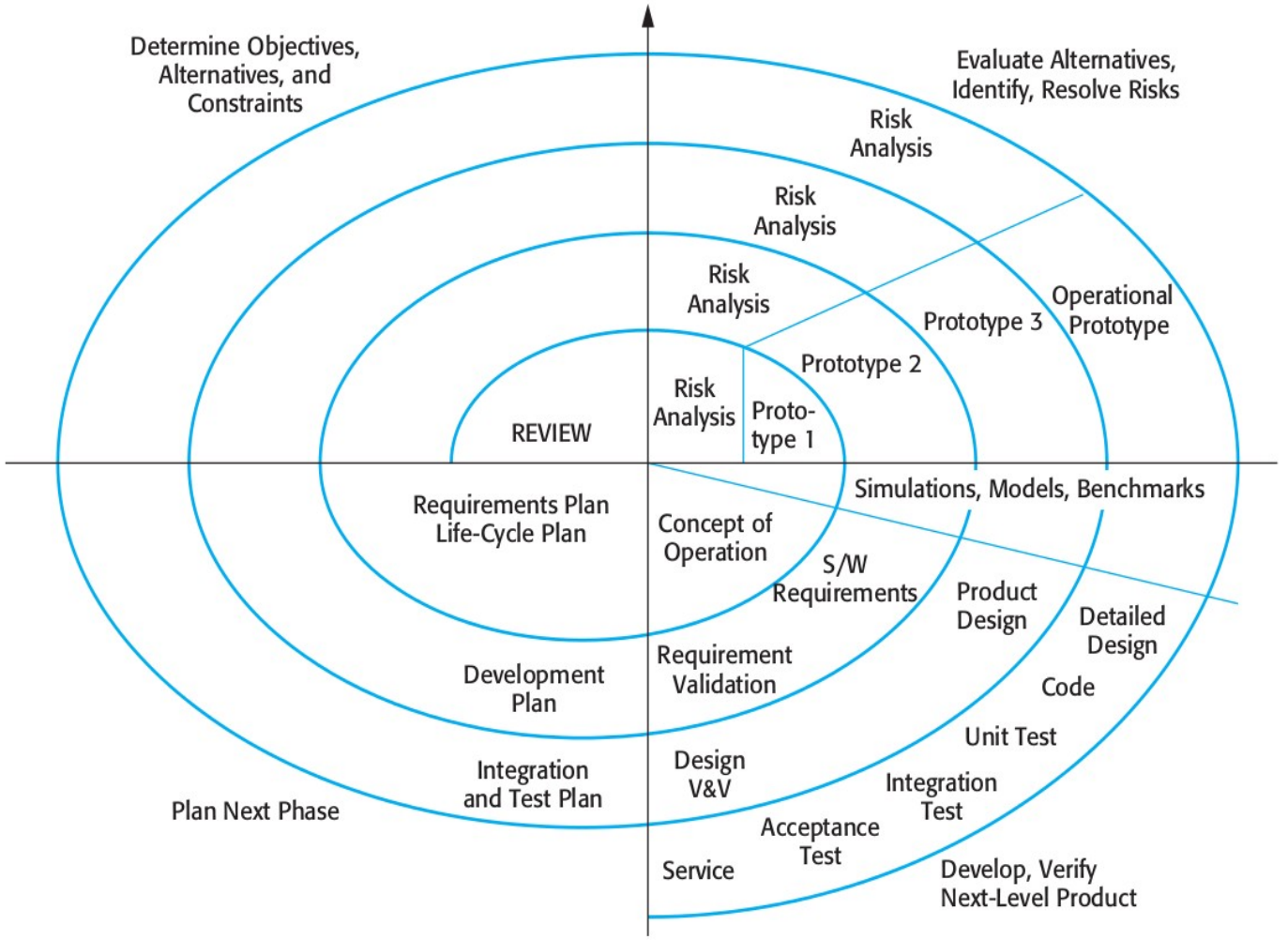
- Birçok avantaj
  - Adımlar prototip gibi kullanılır ancak gerçek ortamda çalıştıkları için sistem tamamlanınca kullanıcılar yeniden öğrenmek zorunda kalmaz
  - Müşteri yazılımı kullanmak için çok uzun süre beklemez
  - Değişikliklere uyum daha kolay olur
  - Öncelikle daha önemli kısımlar teslim edildiğinden çok iyi bir şekilde test edilirler

Artımlı Teslimat

- Bazı dezavantajlar
  - Ortak bileşenlerin belirlenememesi
  - Eski sistem varsa kullanıcıların yeni sisteme sıcak bakmaması
  - Büyük işletmelerin, devlet kurumlarının işin en başından tüm ayrıntıların belirlenmesini ve ona göre kontrat hazırlanmasını istemeleri
- Birden fazla farklı ortamlarda çalışan yazılım grubu bulunan büyük projelere, gömülü sistem geliştirimlerine, güvenliğin çok önemli olduğu ve gereksinimlerin en baştan belirtilmesi gerektiği durumlara pek uygun değildir

Boehm'un Spiral Modeli

- Risk-güdümlü bu model 1988'de ortaya çıkmıştır



Boehm'un Spiral Modeli

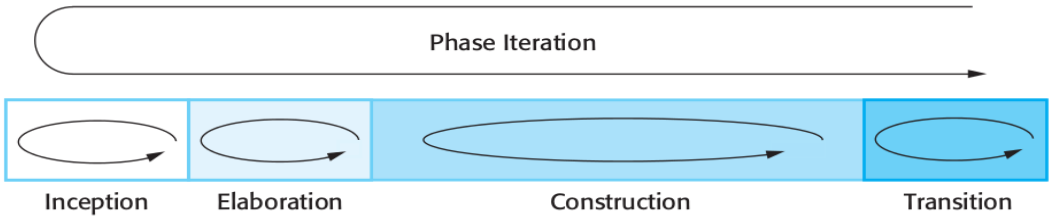
- Değişimi önlemeye ve değişime uyum sağlamaya yönelik bir modeldir
- Spiralin içindeki her döngü dört bölümden oluşur:
  - Hedef belirleme
  - Risk belirleme ve azaltma
  - Geliştirme ve doğrulama
  - Planlama
- Diğer modellerden en önemli farkı risklerin değerlendirilmesi

Rasyonel Birleşik Süreç (The Rational Unified Process)

- UML ve Birleşik Yazılım Geliştirme Süreci üzerine üretilmiş modern bir süreç yönetim modelidir (2003)
- Bilinen modellerin hepsinden parçalar içeren hibrit bir modeldir
- Geleneksel yöntemler sisteme tek açıdan bakarken RUP 3 perspektif sunar:
  - Dinamik, statik ve pratik

Rasyonel Birleşik Süreç (The Rational Unified Process)

- RUP fazları (teknikten ziyade iş dünyası terimleri)
  - Başlangıç (Inception) – Sistem için bir iş modeli ortaya koyulur; kişiler ve sistemle olan ilişkileri; sistem işe ne katkıda bulunacak
  - Detaylandırma (Elaboration) – Detaylı sistem modeli, mimari tasarım, proje planı, ana risk unsurları
  - İnşa (Construction) – Sistem tasarımı, programlama, test; çalışan bir yazılım ve dokümantasyon elde edilir
  - Dönüşüm (Transition) – Canlı kullanıma geçme



Rasyonel Birleşik Süreç (The Rational Unified Process)

- RUP pratik perspektif - iyi yazılım için gerekli altı işlem
  - Artımlı tasarım
  - Gereksinimlerin yönetimi
  - Bileşen tabanlı mimari kullanma
  - Yazılımı görsel olarak modelleme – UML
  - Yazılım kalitesini onaylama
  - Değişiklikleri iyi yönetme

Rasyonel Birleşik Süreç (The Rational Unified Process)

- Gömülü sistemler gibi sistemlere uygun değil
- Yenilikler
  - Fazlar ve iş akışları (iş modelleme, gereksinim tanımı, analiz ve tasarım...) ayrılmış
  - Yazılımın müşteride kullanımı sürecin bir parçası olarak görülüyor
  - Üç genel modelin birleşimi

Önemli Noktalar

- Yazılım süreçleri, bir yazılım sistemi geliştirirken yapılan işlerdir. Yazılım süreç modelleri bu işlerin soyut gösterimleridir.
- Genel süreç modelleri yazılım süreçlerinin düzenini tanımlar. Bu modellere örnekler: şelale modeli, artımlı geliştirim ve yeniden-kullanım tabanlı geliştirimdir.
- Gereksinim mühendisliği, yazılımın tanımını oluşturma sürecidir. Bu tanımlar, müşterinin sistem ihtiyaçlarını, yazılım geliştiricilere aktarmak için kullanılır.
- Tasarım ve gerçekleştirim süreçleri gereksinim tanımlarını çalıştırılabilir bir yazılım sistemine dönüştürme adımıdır. Bu dönüşüm için sistematik tasarım metotları kullanılabilir.
- Yazılım doğrulama, sistemin tanımında olan işleri gerçekleştirdiğini ve kullanıcıların isteklerini karşıladığını onaylama sürecidir.
- Mevcut sistem yeni gereksinimleri karşılamak üzere değiştirildiği zaman yazılımın evrim süreci ortaya çıkmış olur. Değişiklikler süreklidir ve yazılım kullanışlı olabilmek için evrilmelidir.
- Süreçler değişimlerle başa çıkabilmek için gerekli aktiviteleri içermelidir. Gereksinim ve tasarımdaki zayıflıklardan sakınmak için prototip geliştirilebilir. Artımlı geliştirim ve teslim ile değişikliklerin tüm sistemi etkilememesi ve kolaylıkla gerçekleştirilmesi sağlanabilir.
- Rasyonel Birleşik Süreç fazlara ayrılmış ancak aktivitelerini de bu fazlardan ayrı tutan modern ve genel bir süreç modelidir.

Haftaya görüşmek üzere