Dört Temel Yazılım Geliştirme Metodolojisi

I)Yapısal Analiz ve Tasarım 1960 lı yılların sonu 1970 li yıllar

- ❖Fonksiyonel ayrıştırma (functional decomposition) ve veri akış analizi (data-flow analysis) yazılım geliştirme araçlarının temelini oluşturur (modeling tools).

II) Nesneye Yönelik Analiz ve Tasarım (Object-oriented analysis and design) 1980li ve 1990lı yıllar

- ❖Birleştirilmiş Modelleme Dili (Unified Modeling Language-(UML) yazılım geliştirme araçlarının temelidir.

III)Çevik Yazılım Geliştirme (Agile Software Development) 1990lı yılların sonu ve 2000li yıllar

- ❖Yazılım endüstrisinde yazılım ürünü geliştirilirken "lightweight" yaklaşım

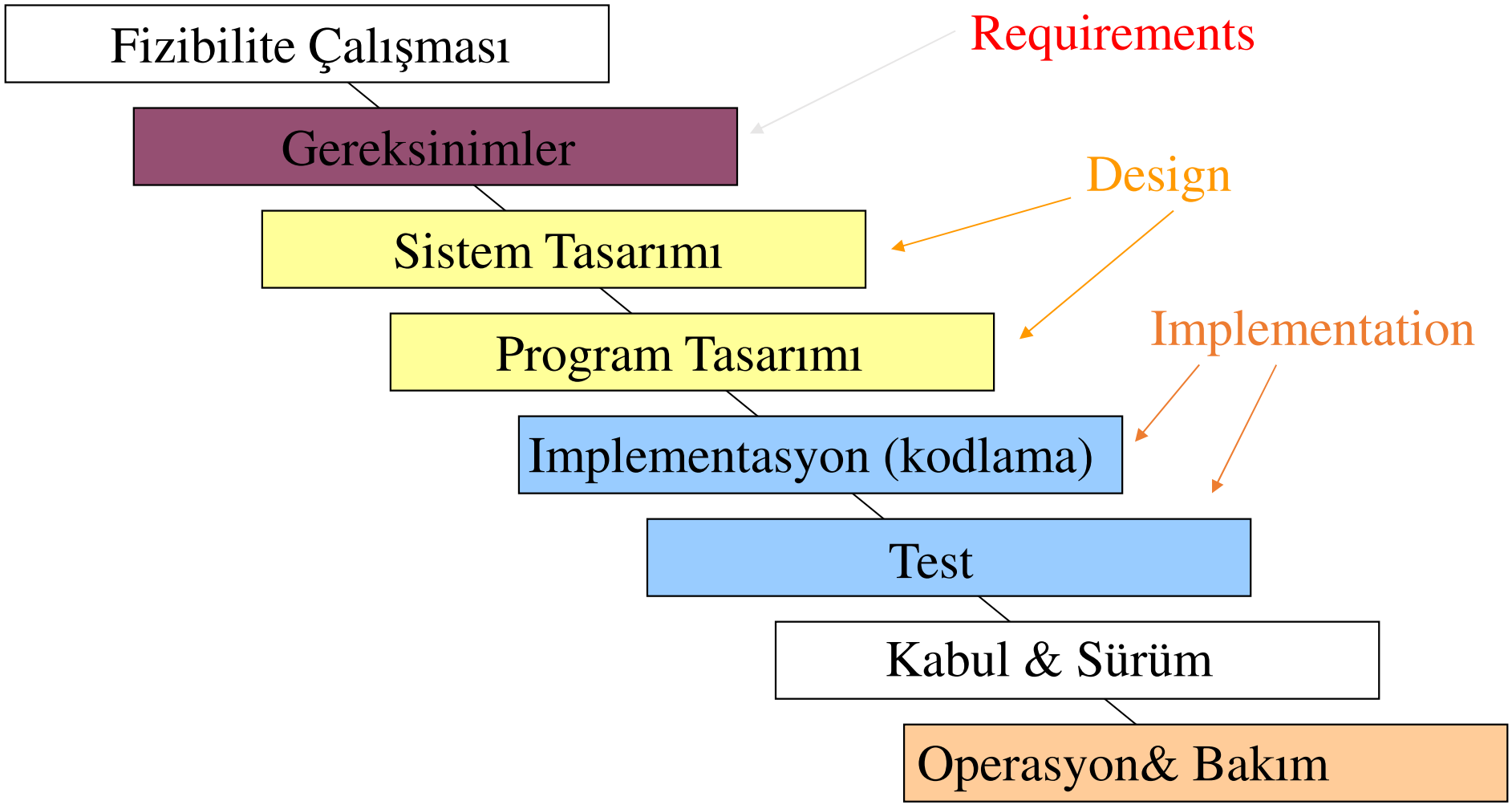
❑ **Alana özel yazılım geliştirme (Aspect-Oriented Software Development) 2000 li yıllar**

- ❖ Diğer yazılım geliştirme metotları terkedilmemiştir; sadece çalışılan alanla ilgili olan metotlar kullanılır.

Her Yazılım Projesinde Hazırlanması Gereken Dokümanlar

- Gereksinimler Dokümanı (Requirements Document)
- Yazılım Projesinin Planlanması Dokümanı (Software Project Planning)
- Yazılım Tasarımı Dokümanları (Software Design Document)
- Testin Planlanması ve Test Raporları (Test Planning and Test Report)
- Final kodu
- Yazılım ürününe ait kullanım kılavuzları (Software Manuals.)

Sıralı Yazılım Geliştirme (Sequential Development)
Şelale Yöntemi (Waterfall Model)



Şelale Modelinin Avantajları

- Kavramsal olarak basittir
- Problemi her biri bağımsız olarak çözümlenecek farklı aşamalara böler
- Yazılım probleminin çözümünde genel bir yaklaşımdır
- Yönetimi kolaydır
  - Herhangi bir aşama tamamlandığında , bu aşamaya ait yapılacakların tümü (work products) gerçekleştirilmiştir.
- Her bir geliştirme aşamasında kalite kontrolü yapılabilir.
- Her bir geliştirme adımında maliyetin kontrolü yapılabilir.

Şelale Modelinin Olumsuzlukları

- Sadece önceden hazırlanmış olan (mevcut olan) dokümanlara bakarak ürünün ayrıntılı çözümüne geçilebilir.
  - Çünkü yazılım ürününe ait tüm gereksinimler tasarım aşamasından önce belirlenmiştir.
- Bu da gereksinimlerin başlangıçta tam olarak belirlenemediği istemler için problem doğurur.
  - Başlangıç aşamasında tüm betimlemeler yapılması mümkün olmayabilir.
- Kısaca gereksinimler değişmez olmalıdır (frozen requirements)

Şelale Modelinin Olumsuzlukları

- Tüm donanım ve teknolojilerin projenin başlangıç aşamasında belirlenmesi gerekir.
- Çok büyük ve pahalı projeler için problemin çözümünün ilk aşamalarında donanım ve ilgili tüm teknolojilerin belirlenmesi problemin çözüm aşamasının sonlarına gelindiğinde problemler yaratabilir.
- Ürünün müşteriye teslimi tüm aşamalar tamamlandıktan sonra gerçekleşecektir.

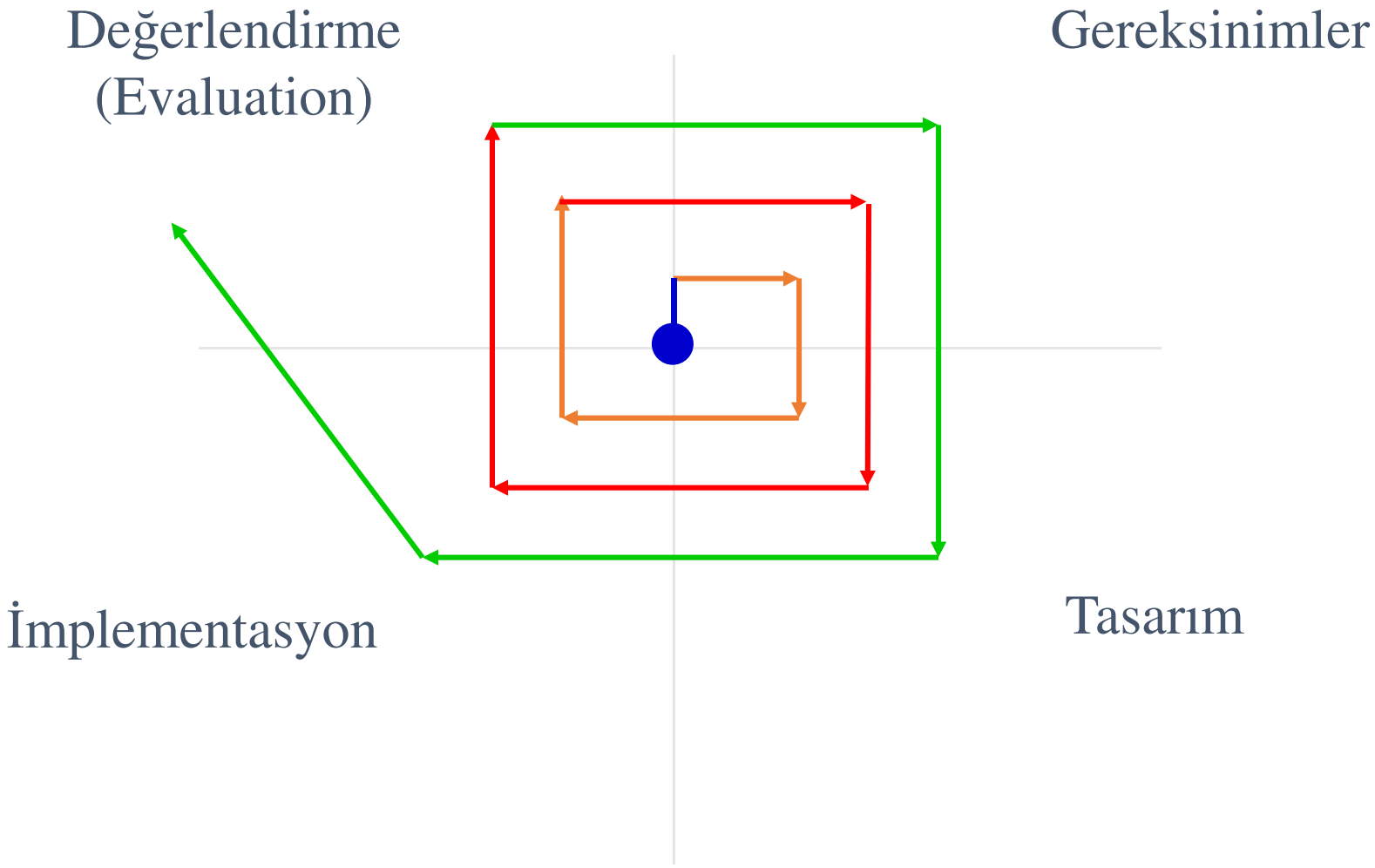
Şelale Modelinin Olumsuzlukları

- Bu model «big bang» yaklaşımını uygular. Müşteriye ürün ya tamamlanmış olarak teslim edilir ya da hiçbir şey teslim edilemez.
  - Kullanıcı proje sonlanıncaya kadar ürün le ilgili hiçbir bilgiye sahip değildir.
- Projenin yarısında parasal sorun yaşanırsa , tamamlanmış hiçbir yazılım ürünü olmayacaktır
- Sonuç olarak bu yöntem doküman odaklıdır. Her aşamanın sonunda dokümantasyonu olmasını gerektirir.

Şelale Yöntemi Hangi problemlere Daha uygundur?

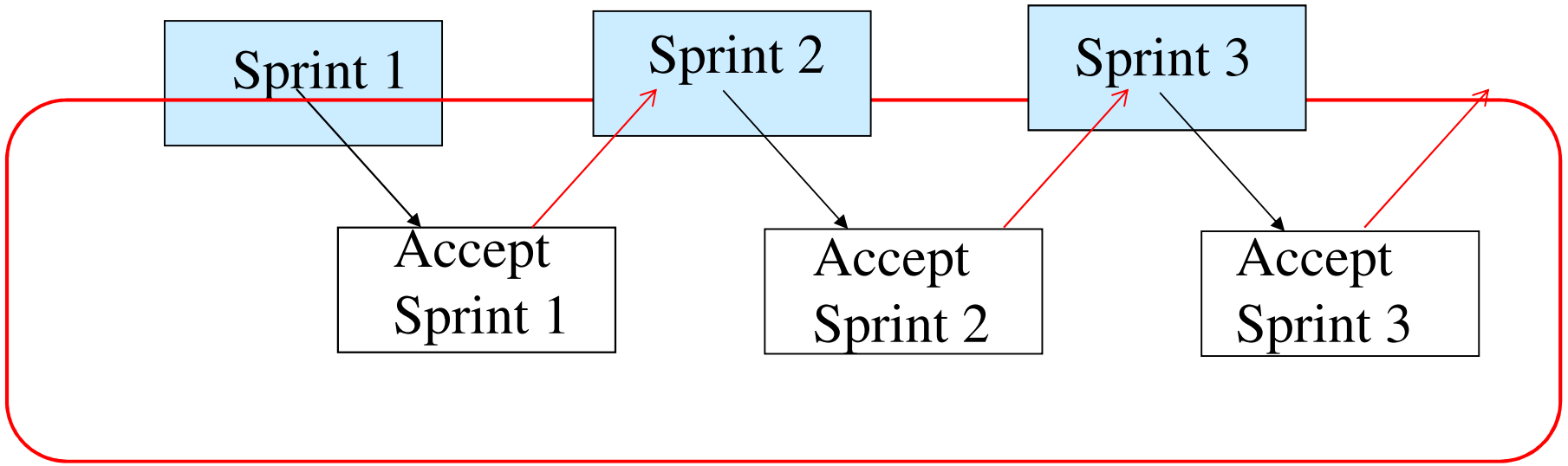
- Gereksinimleri iyi tanımlanmış (well-defined) olduğu projeler için daha uygundur.
- Gereksinimlerin kolaylıkla belirlenebildiği ve teknolojik kararların çok kolay alınabildiği projeler için çok uygundur.
- Ürün geliştirme takımı problem alanında deneyimli ise, gereksinimler açık olarak belirlenmiş ise en uygun geliştirme yöntemi olarak alınabilir.

Iteratif Yenileme (Iterative Refinement)



Artırımsal Geliştirme (Incremental Development) (Sprints)

- Ürün geliştirme takımı yazılımın tüm yaşam döngüsü boyunca (analiz-tasarım-implementasyon-kod) her bir artırım için (sprint) çalışır
- Her bir «sprint» belli bir zaman diliminde tamamlanmalıdır (örneğin 4 hafta).
- Bir «sprint» büyüklüğü yazılım takımının büyüklüğüne başlıdır. 5-10 kişi olabilir).



Proseslerin Sıralanışı

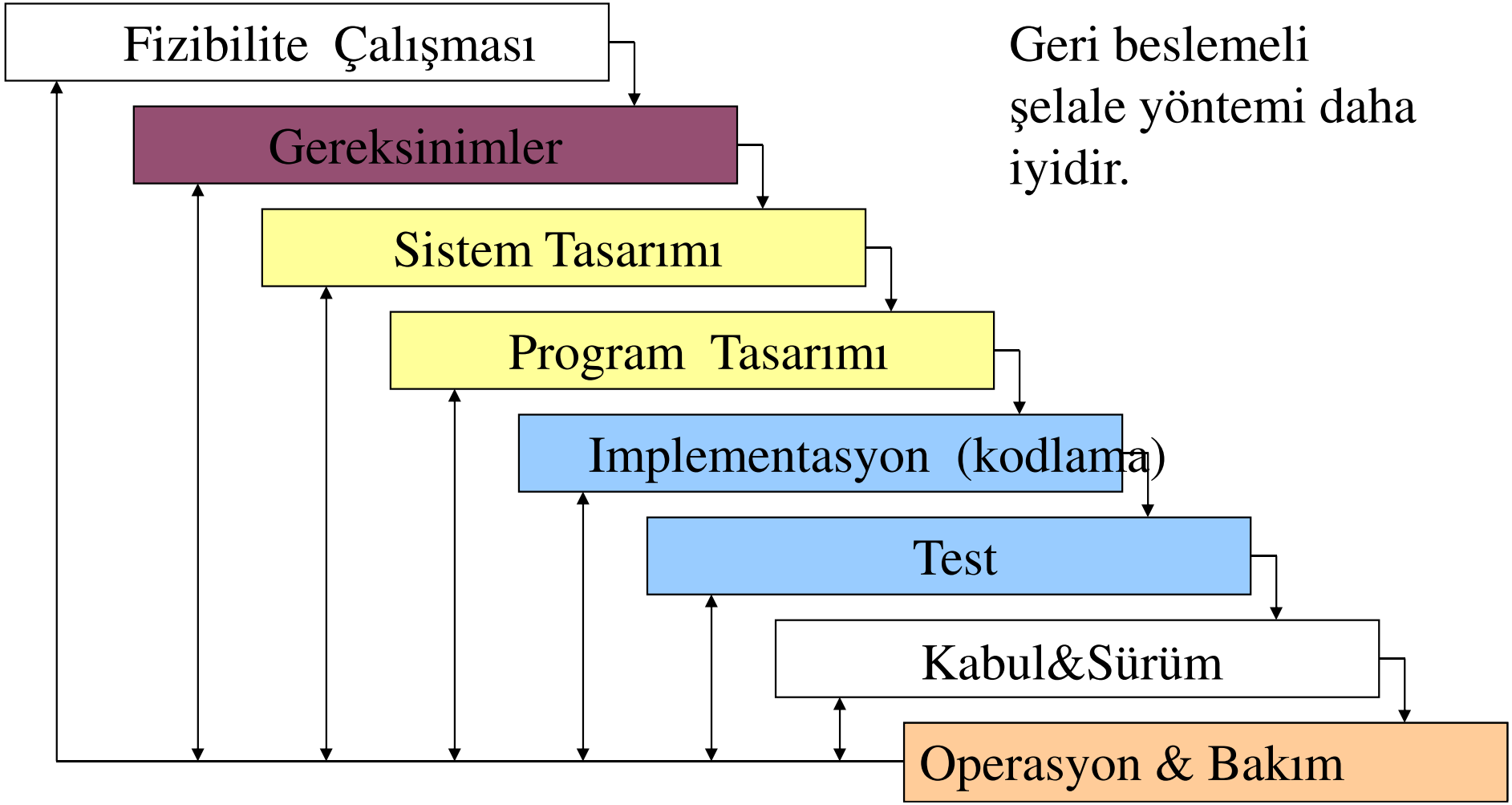
Tam olarak sıralı bir model mümkün değildir.

Örnekler:

- Fizibilite çalışması önerilen bütçeyi belirleyemez ve gereksinimlerle ilgili bir ön çalışma olmadan ve geçici bir tasarım olmadan proje ile ilgili zaman çizelgesi oluşturamaz
- Ayrıntılı tasarım ile implementasyondaki açıklar gereksinimler betimlemelerindeki belirsizliklerden oluşur.
- Gereksinimler ve mevcut teknoloji zamanla değişebilir.

Bu nedenle planlama iterasyona uygun olarak hazırlanmalıdır.

Değiştirilmiş Şelale Yöntemi (Modified Waterfall Model)



Geri beslemeli şelale yöntemi daha iyidir.

Iteratif Prosesler
Gereksinimler ve Risk

- **Gereksinimlerdeki hatalar düzeltilmesi en pahalı yanlışlardır.**
- Gereksinimlerin sağlanıp sağlanmadığı operasyonel bir sistem gerçekleştirilene kadar, özel olarak kullanıcı arayüzleri ile anlaşılmaz.
- Hızlı bir şekilde çevrimiçi (online) bir sistemin oluşturulması ve bunun kullanıcılarla test edilmesi, gereksinimlerin anlaşılabilirliğini arttıracaktır.
- Örneğin, karmaşık bir uygulamada başlangıç sürümlerinin oluşturulması (Start-up time of launching)

Çevrimiçi Sistemlerde Artırımsal Geliştirme

- ❑ Geliştirilen yazılım çevrimiçi olarak sürüldüğünde ürünü küçük artırımlarla bölerek geliştirmek ve ardışık hızlı sürümlerle geliştirmek mümkündür.
- ❑ Bu yaklaşım kullanılan uygun mimari ile sistemin sürekli olarak iyileştirilmesi için mükemmeldir.
- ❖ Gömülü sistemler (Embedded Systems) uygun değildir.
- ❖ «shrink wrapped» , diğer bir kullanımı ile «off the shelf» yazılımlar için uygun değildir.
- ✓ Standart yazılım uygulamalarıdır. Satın alındığında kullanım koşulları kabul edilmiştir.

Karma Prosesler (Mixed Processes)

Pratikte pek çok geniş kapsamlı projenin oluşturduğu proses spesifik bir geliştirme için uygundur. Örneğin:

- ❖ Herhangi bir sistem tek tek proses adımlarının gerçekleştirildiği sıralı bir prosesi iterasyon ile birlikte kullanabilir. Bu yönteme bazı uygulamalarda **spiral** geliştirme prosesi adı da verilmektedir.
- ❖ Kullanıcı arayüzlerinin kullanıcılar tarafından test edilmesi gerekir.
- ✓ Bu da temelde işlemlerin sıralı gerçekleştirildiği bir süreç olmasına rağmen tekrarlanan (iterative) bir geliştirmeyi zorlar.

•Örneğin aşama 1 de sistem yineleme (iteration) ile geliştirilebilir ve sıralı olarak geliştirilmiş aşama 2 için gereksinimler olarak kullanılır.

Karma Proses Örnekleri

Tekrarlamalı Gelişme (Iterative Refinement) ve Şelale Modeli

Bir grafik paketinin programlama ortamına ilave edilmesi

Aşama 1 : Tekrarlama (Iterative refinement)

Mevcut çalışma ortamı (örneğin C++)bir ön işlemci ve çalışma zamanı (run-time) destek paketi ile genişletilir. Kullanıcılarla test edilir. Kullanıcılar fonksiyondan memnun olana kadar yeri versiyonları yapılır. Kod her sefer atılır (**Throw the code away).**

Aşama 2:Değiştirilmiş Şelale Yöntemi

Gereksinimlerin biçimsel bir dizilişi için Aşama 1' in sonucu temel olarak kullanılır. Yeni derleyici yazılır ve çalışma zamanı sistem grafik elemanları ile birleştirilir. İhtiyaç oldukça gereksinimlerde küçük düzeltmeler yapılır.

Birleşik Prosesler

Büyük yazılım geliştirme organizasyonlarının gereksinimleri için tasarladıkları kendilerine ait tasarımları mevcuttur. Örneğin:
Amazon.com pek çok yazılım geliştirme aşamalarını dört haftada tamamlanacak şekilde böler.
Lockheed Martin US hükümetinin yazılım sözleşmelerinin yönetimi için uyulması gereken prosesleri izler.

SAP (iş yazılımı -business software) ticari müşterilerin işlerinde fonksiyonelliği gerçekleştirmelerini sağlar.

Microsoft çok çeşitli aygıtların testine ve geriye (geçmişe yönelik) uyumluluğuna (backward compatibility) büyük önem verir.

Yazılım Süreçlerinde Modern İlerlemeler

- Yazılım geliştirme prosesleri süresince yapılan değişiklikler oldukça pahalıdır.
- Gereksinimler iyi anlaşılmamış ise veya değişmeye açık ise esnek bir yazılım geliştirme prosesinin seçimi uygundur (iteratif geliştirme, sprints, aşamalı implementasyon gibi)
- Büyük yazılım sistemlerinde birbirleri ile çok fazla ilişkisi olan bileşenler vardır. Bu nedenle ürünün geliştirilmesi süresince tasarımda büyük değişiklikler yapılması tercih edilmez (Değiştirilmiş Şelale Yöntemi gibi Sıralı Proses)
- Geliştirilen yazılım ürünü için piyasa şartları iyi anlaşılmış değil ise, arttırılmış (incremental) prosesin kullanılması uygundur .Böylece yazılımın operasyonel olarak müşterilerin önüne mümkün olduğu kadar hızlı olarak çıkması sağlanmış olur.

Yazılım Prosesleri ile İlgili…

Tamamlanmış projelerin temel proses adımları olacaktır. Fakat geliştirme prosesi her zaman kısmi gelişmeye açıktır.

Herhangi bir prosesi seçmek nedeni ile risk taşınır. Fakat bu riskler aşağıdakiler göz önüne alınırsa azaltılabilir:

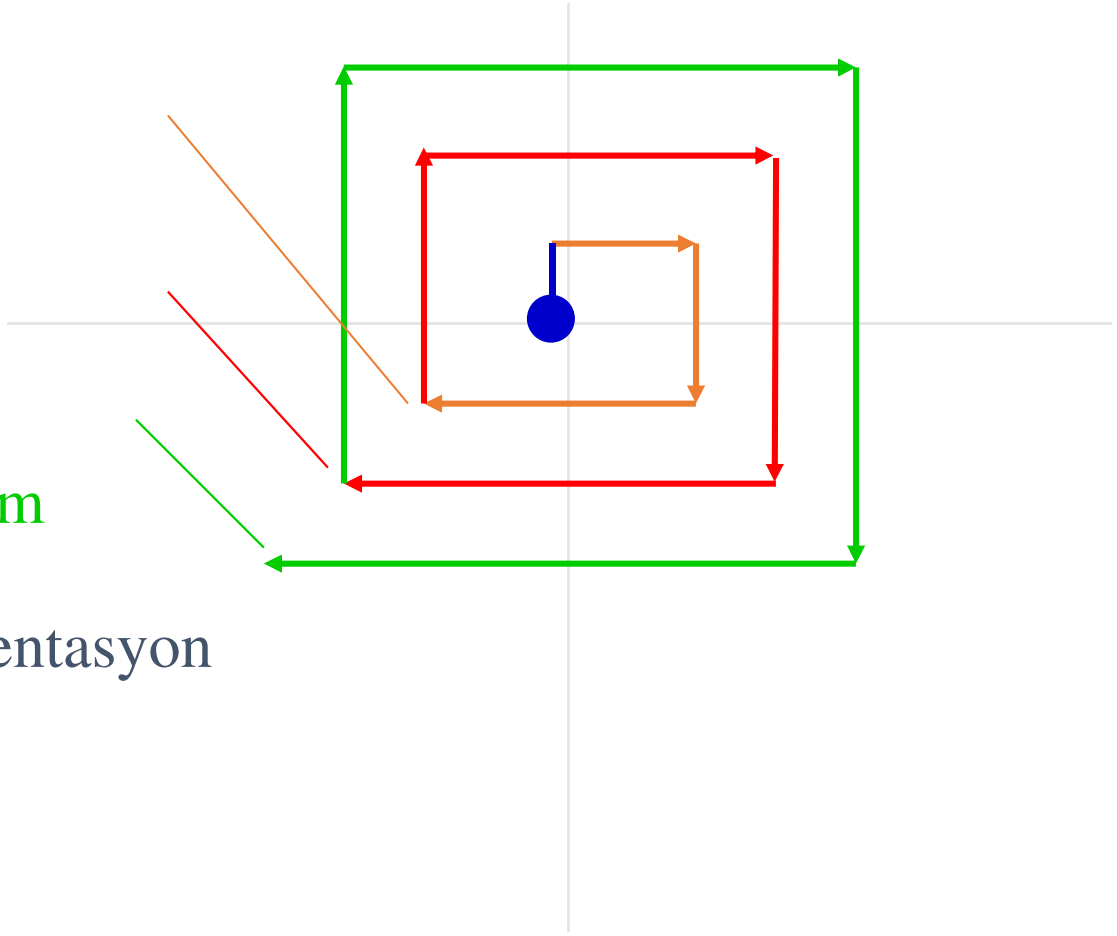
- Anahtar bileşenler için prototipleme yapılması
- Sık sık sürüm yapılması (büyük projeleri parçalayarak)
- Müşteriler ve kullanıcılar ile projenin başından itibaren ve sık sık test yapılması
- Belli, aşikar olan (visible) yazılım prosesinin izlenmesi
- Yeniden kullanılabilir bileşenler oluşturulması

Iteratif Süreç Modeli

Değerlendirme

Gereksinimler

İlk sunum



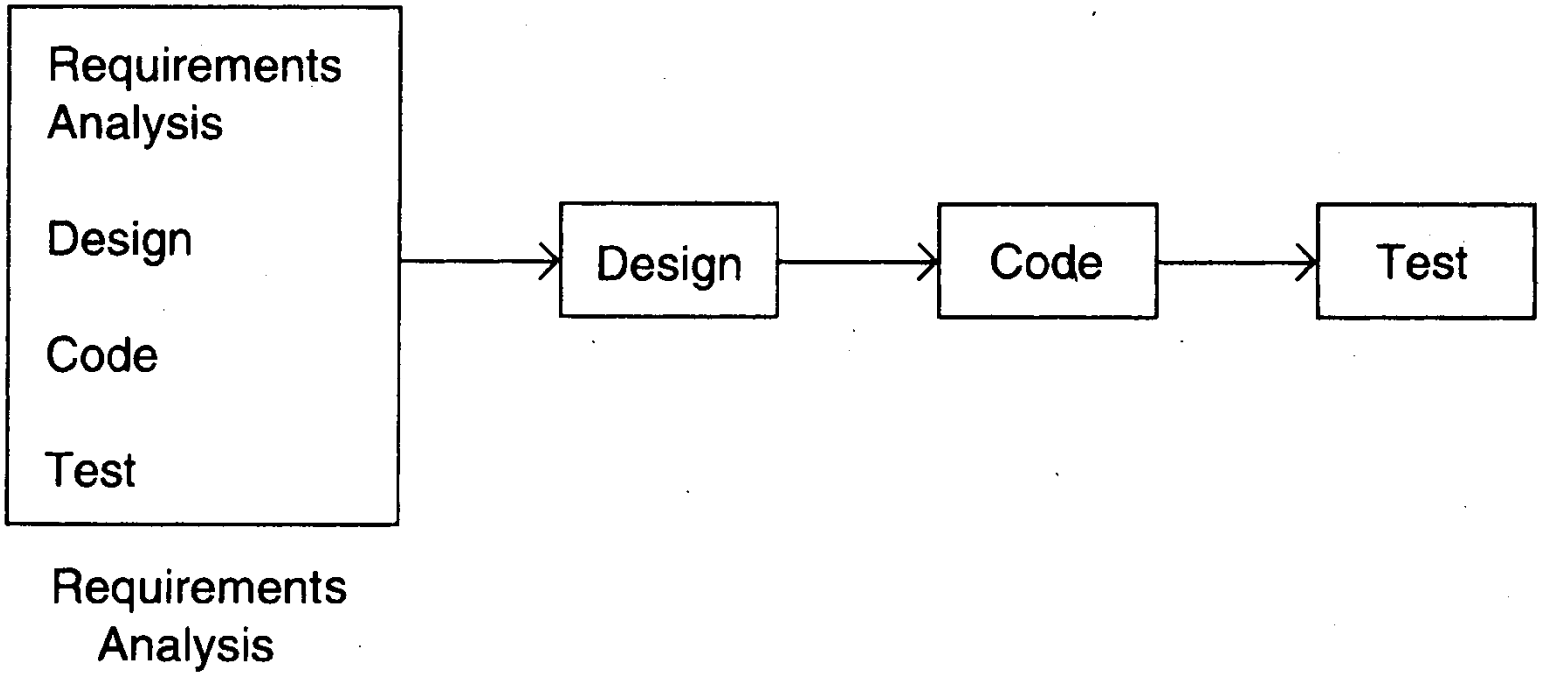
İkinci sunum

Üçüncü sunum

Implementasyon

Tasarım

Prototipleme Süreç Modeli



- Gereksinimler değişime daha fazla açıktır. Yine de her aşamada değiştirilmesi mümkün değildir.
- Prototipleme modeli tümüyle bilinmeyen gereksinmeler için çözüm için projenin başlangıcında seçilmiş olabilir.

Prototipleme Süreç Modeli

- Proje karmaşık bir yapıda, oldukça geniş kapsamlı ise , mevcut bir sistem yoksa ya da mevcut bir sistem olup ta işlemleri ile ilgili hiçbir bilgi yoksa prototipleme ile yazılım geliştirme yöntemi uygun bir seçim olabilir.
- Gereksinimler açık olarak ifade edilememiş ise , belirlenen gereksinimlerin implementasyonunu gerçekleştirecek bir algoritma geliştirilir .
- Prototipleme modeli gereksinimlerin riskini azaltır. Böylece gereksinimlerin belirtimi (specification) durumunda da risk azalacaktır.

Sonuç olarak:

- Prototipleme Süreç Modeli gereksinimler aşamasının yer değiştirdiği küçük bir şelale yöntemidir