YAZILIM TESTİ VE PROJE YÖNETİMİ

22 Nisan 2024

Yazılımın Yaşam Döngüsü Sürecince Test

- «Sequential» Geliştirme Modelleri
  - Waterfall
  - V-Model
- «Iterative/Incremental» Geliştirme Modelleri
  - Iterative
  - Incremental
  - Iterative-Incremental
  - Agile Software Development Model
  - Continuous Integration and Continuous Deployment

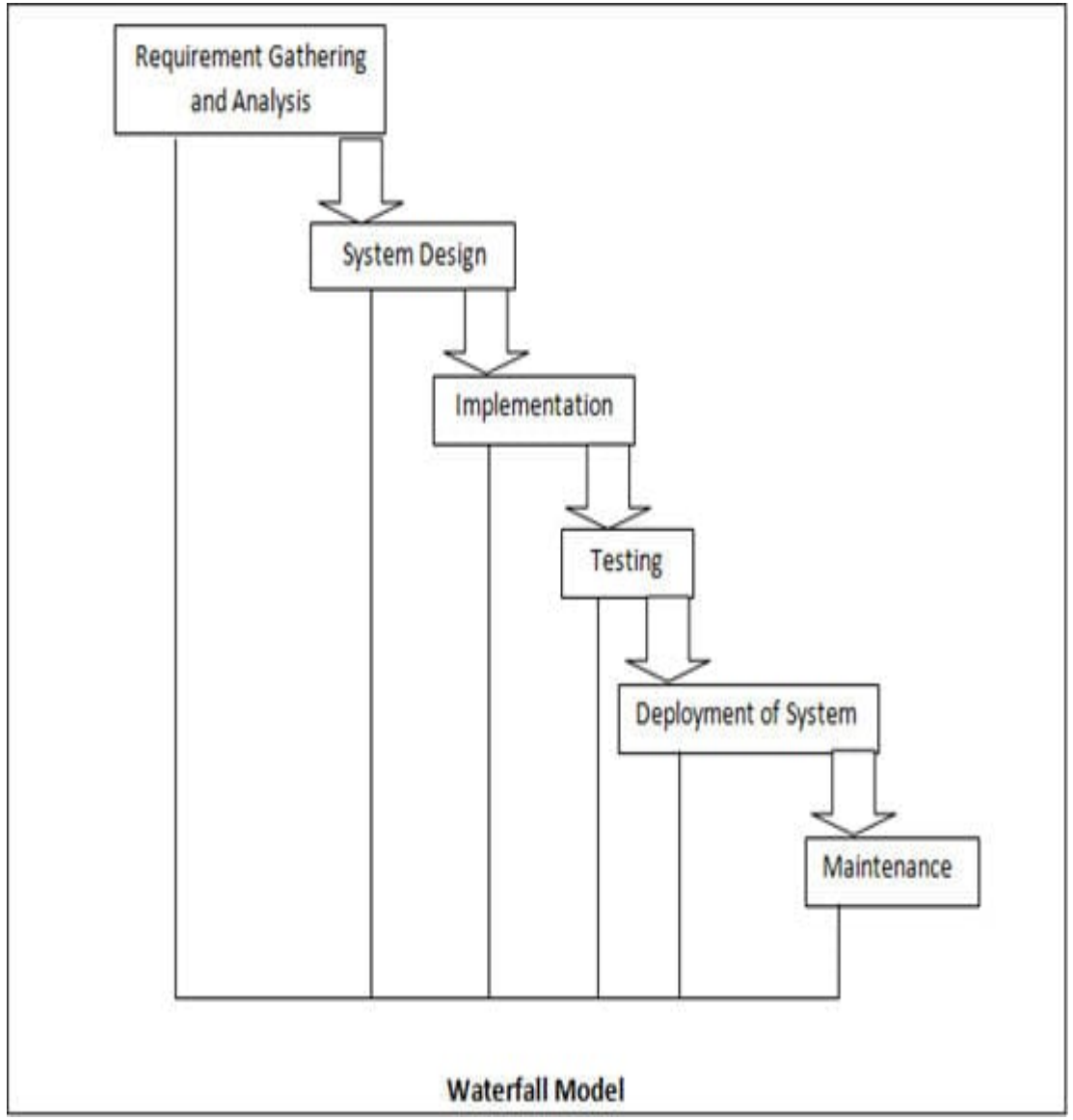
Şelale (Waterfall) Yöntemi

1970 yılında Winston Royce tarafından geliştirilmiş temel yaşam dönfgüsü modelidir.

Sıralı olarak çoklu sürecin izlenmesini gerektirir.

Bu yaklaşım gereksinimlerin bilinmesi (well known) durumunda yararlı sonuçlar verir.

Ayrıca teknolojinin de tanımlanmış ve kaynakların sağlanmış olması gerekir.



Şelale Yöntemi ve Test Süreçleri

Requirement Gathering and Analysis: Tüm gereksinimler belirlenmiş, analiz edilmiş ve tanımlanmıştır. Test edilebilir ve test edilemez olma durumlarına karar verilmiştir.

System Design: Gereksinimler analizi üzerine doküman odaklı tasarım gerçekleştirilir. Donanım ve yazılım gereksinimleri tanımlanır.

Implementation: Bileşenlerin her birine ilişkin en iyi (güçlü/ **robust)** kod yazımı gerçekleştirilir ve gerekli entegrasyonlar gerçekleştirilir

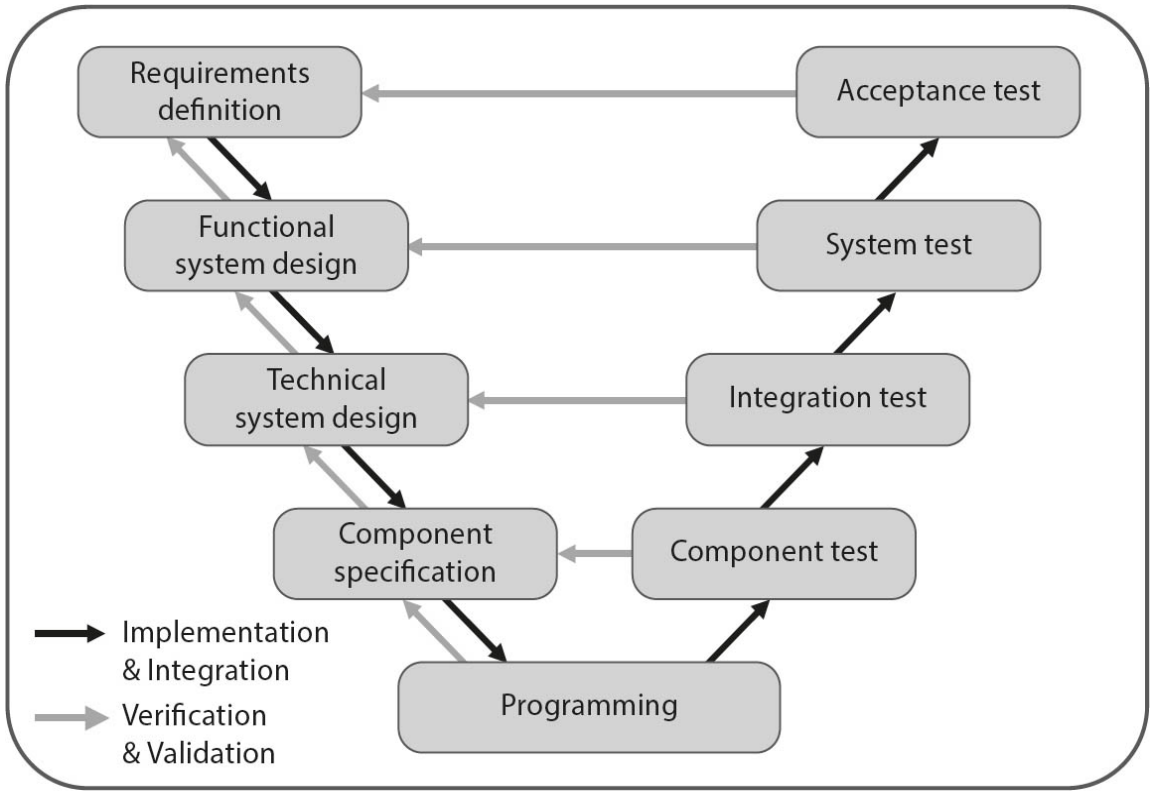
System Testing: Birleştirilmiş bileşenler (**integrated componenets**) tüm sistemi verir ve bunlar her bir gereksinime karşılık gelmektedir. Aynı zamanda test süreçlerinin de raporları alınmaktadır.

System Deployment: Sistemin **zero bugs** ile kararlı (**stable**) olup olmadığına karar verilir. Bunun için tüm test kriterleri karşılanmıştır, çevre ile nasıl iletişim kurulacağı **Environment Setup** gerçekleştirilmiştir.

System Maintenance: Gereksinimlerin etkin olarak, yani istenildiği şekilde karşılandığı, çalıştığı ortamla uyumlu (**suitable environment**) bir uygulama geliştirildiğinin görülmesi.

Bu aşamada herhangi bir **defect** ile karşılaşılması durumunda bu sabitlenmeli (**fixed**) ve çevre ile uyumlu olarak güncellenmelidir (**deployed /updated**)

Geleneksel Yazılım Geliştirmede V-Modeli



1. Aşamalar V-şeklinde ve sıralı olarak gerçekleştiği için Şelale Modelinin gelişmişidir.

2.Aynı zamanda «Verification and Validation Model» olarak ta adlandırılır.

3.Geliştirme aşamasının her birine karşılık gelen bir test vardır.

4. Her test düzeyi V&V testlerini içerebilir.

4.1 **Did we build the system right?** sorusunun cevabı **Verification**

4.1.1 «Test objects» ilgili geliştirme aşamasında doğru olarak ve istenilenleri sağlayacak şekilde geliştirildi mi?

4.1.2 İlgili adımın çıktısı sonraki girdi için istenilenleri sağladı mı?

4.2 Did we build the right system? *sorusunun cevabı* ***Validation***

4.2.1 «Test object» hedeflenen içerik midir? Gerçekten geliştirilmesi istenilen midir?

4.2.2 Test takımı, test nesnesinin sorunu gerçekten çözüp çözmediğini ve amaca uygun olup olmadığını kontrol eder.

V-Modelin Sınıflandırması
Verification Phase- Coding Phase- Validation Phase

a) Verification Phase (Doğrulama Safhası):

Business Requirement Analysis: Müşteriyle iletişim kurularak gereksinimlerin anlaşılması

System Design: Yazılım ve donanım gereksinimlerine göre sistemin ve bileşenlerinin tasarımı

Architectural Design: Yüksek-düzey (high-level) tasarımdır, mimari betimlemeler gerçekleştirilir.

Module Design: Düşük düzey (low-level) tasarımdır, tüm modüllerin ayrıntılı iç tasarımıdır.

b)Coding Phase:

Geliştirme aşamasındaki kodun içerildiği bölümdür. Programlama dili ve belirlenmiş olan mimari tasarım hayata geçirilir.

c) Validation Phase (Sağlama safhası):

Unit Testing: «bug» ları erken aşamalarda yok etmek üzere tek tek modüllere uygulanır.

Integration Testing: Sistemdeki farklı modüller arasındaki iletişimin test edilmesidir.

System Testing: Tüm sistemin testidir.

Acceptance Testing: Business requirements ile ilişkilidir. Kullanıcının bakış açısından değerlendirilir.

Yazılım Sisteminin Tamamlanması için Temel Testler

Component test

Tek tek her bileşenin gereksinimleri sağlamasıdır.

Integration test

Bileşen gruplarının ilgili teknik tasarım doğrultusunda etkileşimlerinin sağlanmasıdır.

System test

Sistemin bir bütün olarak belirtilen gereksinimlerine göre çalışmasının sağlanmasıdır.

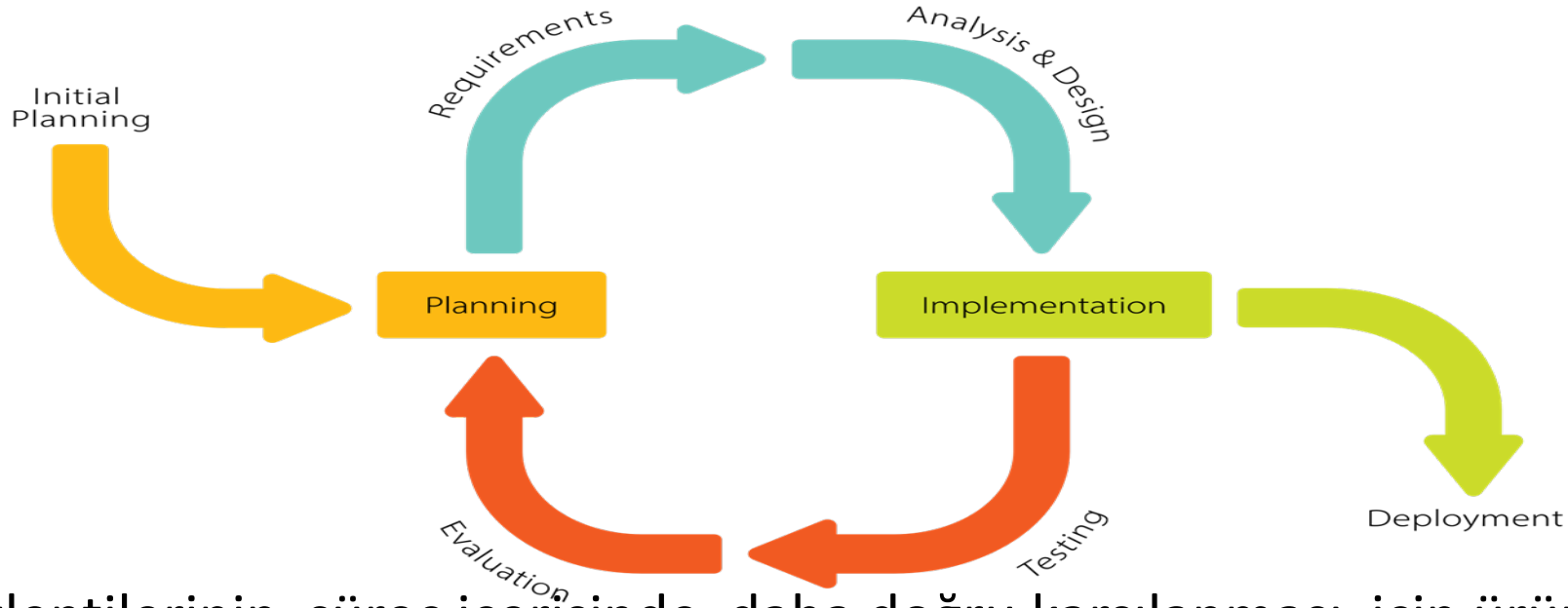
Acceptance test

Sistem bir bütün olarak kontrol edilir, bu müşteri ve son kullanıcı gereksinimleri için verilen bildirimlerin sağlanıp sağlanmadığının kontroludur.

Test için niçin V-Model?

- Geliştirme ve test süreçleri ayrılmıştır. Her biri aynı oranda önemlidir.
- Modelin “V” şekli, testin doğrulama/onaylama (verification/validation) yönlerini görselleştirmiştir.
- «Collabarative test» seviyeleri arasında ayrım yapar
  - Her seviyede test ilgili geliştirme seviyesine göre gerçekleşir.
- «Early Testing» yaklaşımı geçerlidir.
  - « Late testing» gibi görünmüyor mu?
- Avantaj ve dezavantajları Şelale Yönteminin bu özelliklerine yakındır.

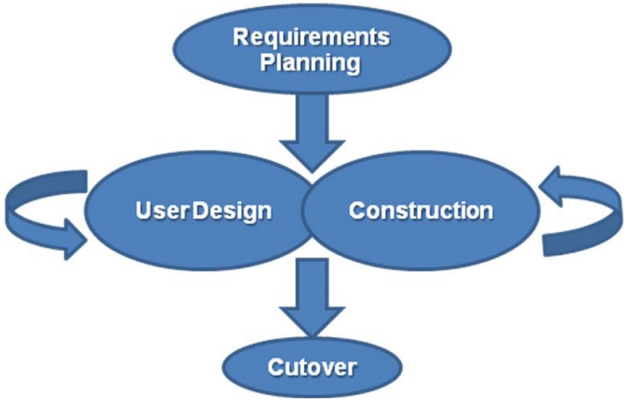
Iterative and Incremental Development
İteratif ve Artırımsal Ürün Geliştirilmesi



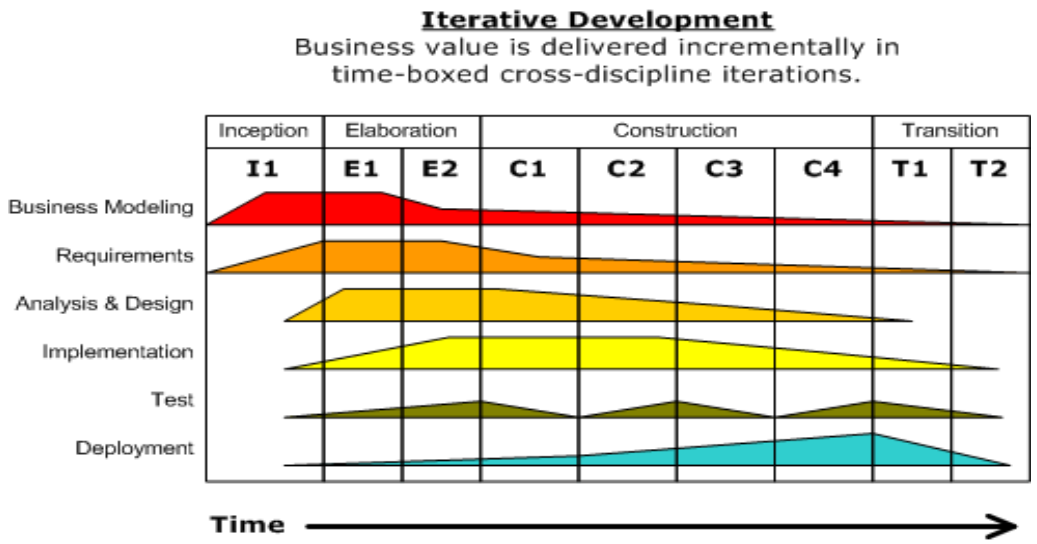
- Müşteri beklentilerinin süreç içerisinde daha doğru karşılanması için ürünü adım adım iyileştirmek hedeflenir.
- Önceki geliştirme aşamalarından edinilen deneyim, gerçek dünya ve önceki sistem sürümlerinden gelen müşteri geri bildirimleriyle birlikte değerlendirilir.
  - Bunlar gelecekteki yinelemelerde geliştirmek için kullanır.
- Mevcut durumda iyileştirmeler, hata düzeltmeleri veya belirli özelliklerin değiştirilmesi, genişletilmesi veya eklenmesi mümkündür.

Iterative /Incremental Development

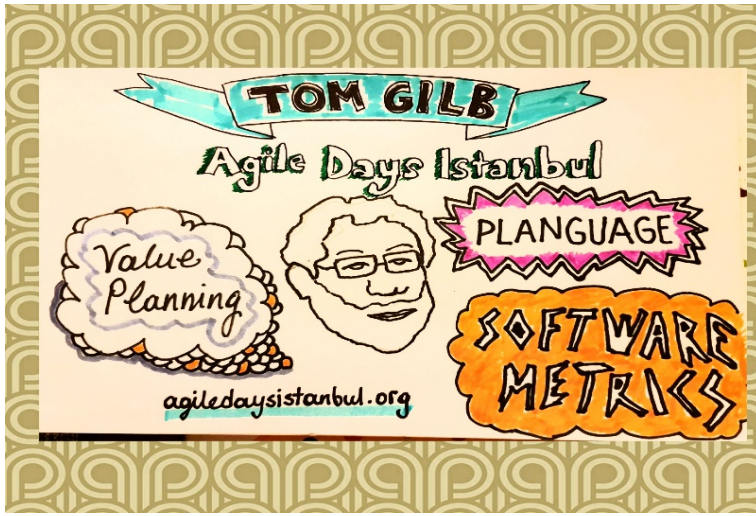
- Spiral model - Boehm - 1986
- Rapid Application Development (RAD) - James Martin 1991
- Rational Unified Process (RUP)- Phillip Kruchten 2003
- Evolutionary Development - Tom Gilb 2005



RAD

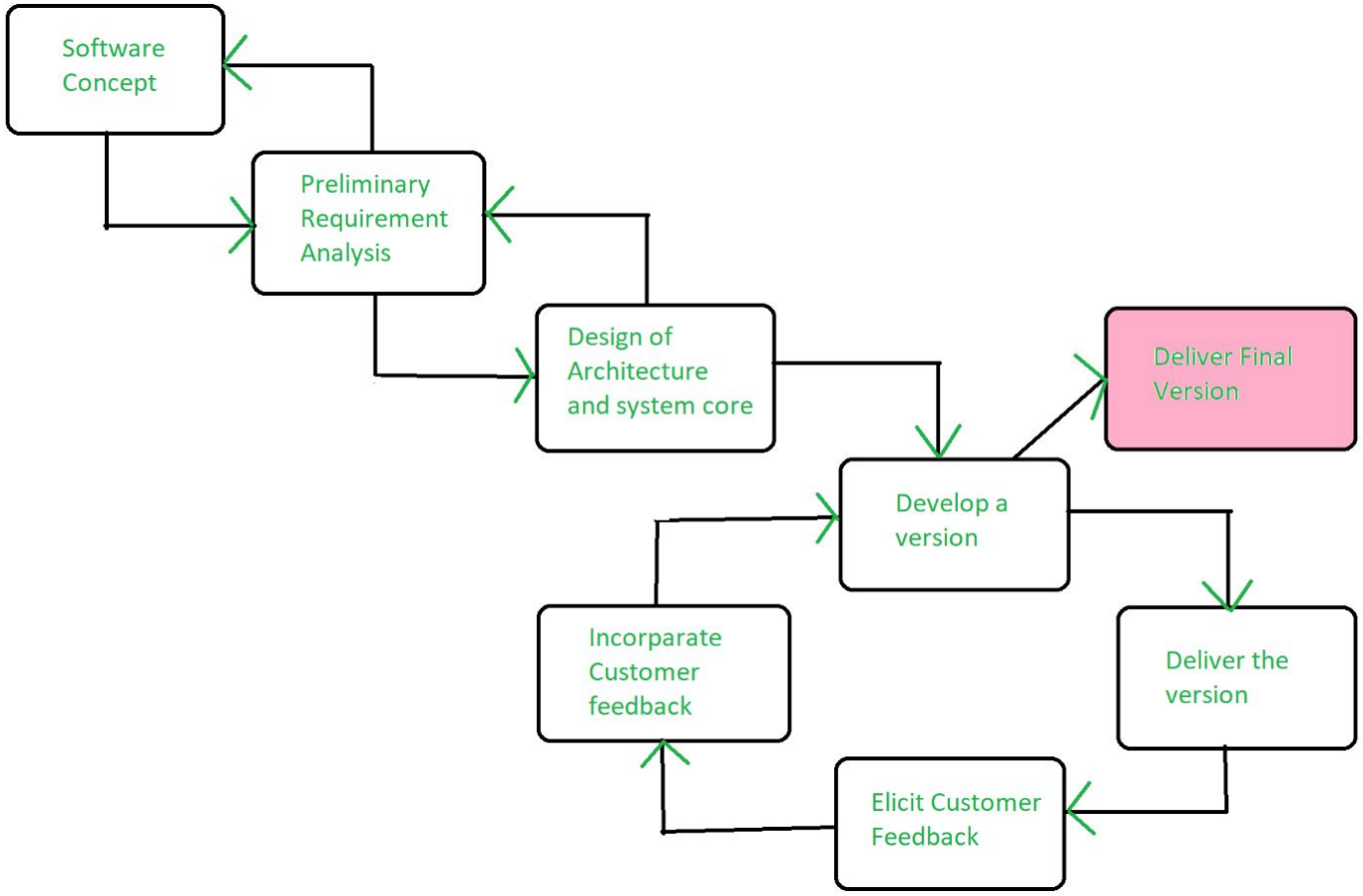


RUD



2018

Evolutionary (Evrimsel) Model



1. Bir sonraki döngünün planlanması için kullanıcılara bir geri bildirim sağlanır ve geliştirme ekibi de ürünün planı veya sürecini değiştirerek bunlara yanıt verir.
2. Bu nedenle yazılım ürünü zamanla gelişir.
3. İteratif ve artırımsal modellerin dezavantajı, projenin başlangıcından teslimine kadar sürenin uzun olmasıdır.
4. Evrimsel model, işin daha küçük parçalara bölünmesini, bunlara öncelik verilmesini ve daha sonra bu parçaların müşteriye tek tek teslim edilmesini önerir.
5. Parça sayısı çok büyüktür ve bunlar müşteriye teslim edilecektir.
6. Modelin avantajı, projenin başlangıcından itibaren müşterinin gereksinimlerinin sürekli olarak sağlandığı (verify) ve doğrulandığı (validate) ölçülebilir hizmetler alarak güveninin artmasıdır.
7. Model, değişen gereksinimlere izin verir ve ürün sürdürülebilir (bakım aşamasının gerçekleştirileceği) iş parçalarına bölünür.

Evrimsel Modelin Uygulamaları

- Büyük projeler için uygundur. Zira artırımsal implementasyon gerçekleştirilir.
- Evrimsel model ilke olarak yazılımın tüm özelliklerinin tamamlandığını beklemek yerine çekirdek özellikleri ile başlayarak ilerlemeyi hedefler.
- Evrimsel model nesneye yönelik yazılım geliştirme ile uygulanabilir.
  - Zira sistem nesneler üzerinde kolaylıkla parçalanabilir.

Evrimsel Modelin Uygulanması ile ilgili Koşullar

- Kullanıcı /müşteri gereksinimleri açıktır ve geliştirme takımına ayrıntılı olarak aktarılmıştır.
- Bağımsız parçalar üzerinde çalışırken gereksinimlerin küçük oranlarda değişmesi söz konusudur.
  - Modelin geliştirilmesi zaman gerektiğinden piyasa kısıtları için bir miktar süre kalması gerekecektir.
- Model uygulanırken alınan riskler yüksektir hedeflere sürekli olarak ulaşabilmek için, müşteriye de raporların sürekli gönderilmesi gerekecektir.
- Yeni bir teknoloji üzerinde çalışılmakta iken öğrenmenin zaman gerektirmesi nedeni ile bu model tercih edilebilir.

Evrimsel Modelin Avantajları

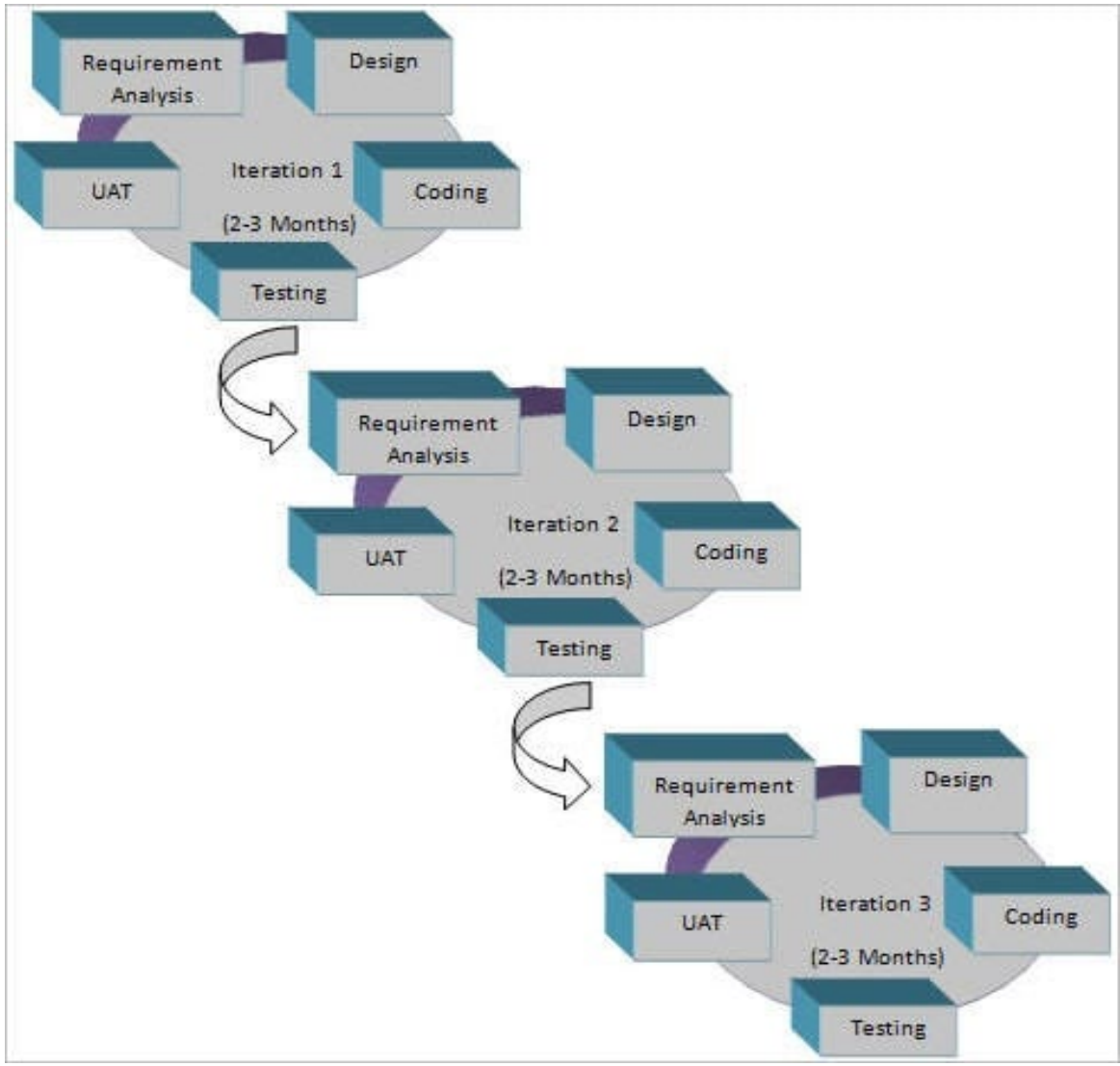
- Değişen gereksinimlere uyum sağlayabilir.
- Artımlı geliştirmeyle fonksiyonel bileşenler veya prototipler erken teslim edilebilir.
  - Böylece daha hızlı kullanıcı memnuniyeti ve geri bildirim elde edilebilir.
- Sürekli kullanıcı girdisi ve katılımına izin verilir.
  - Bu, yazılımın kullanıcının beklentilerine uymasını ve ihtiyaçlarını içermesini sağlar.
- Karmaşık görevler, etkili bir şekilde yönetilebilir.
  - Projenin daha küçük, yönetimi daha kolay bölümlere ayrılmasıyla geliştirme süreci daha basitleşir.

Evrimsel Modelin Dezavantajları

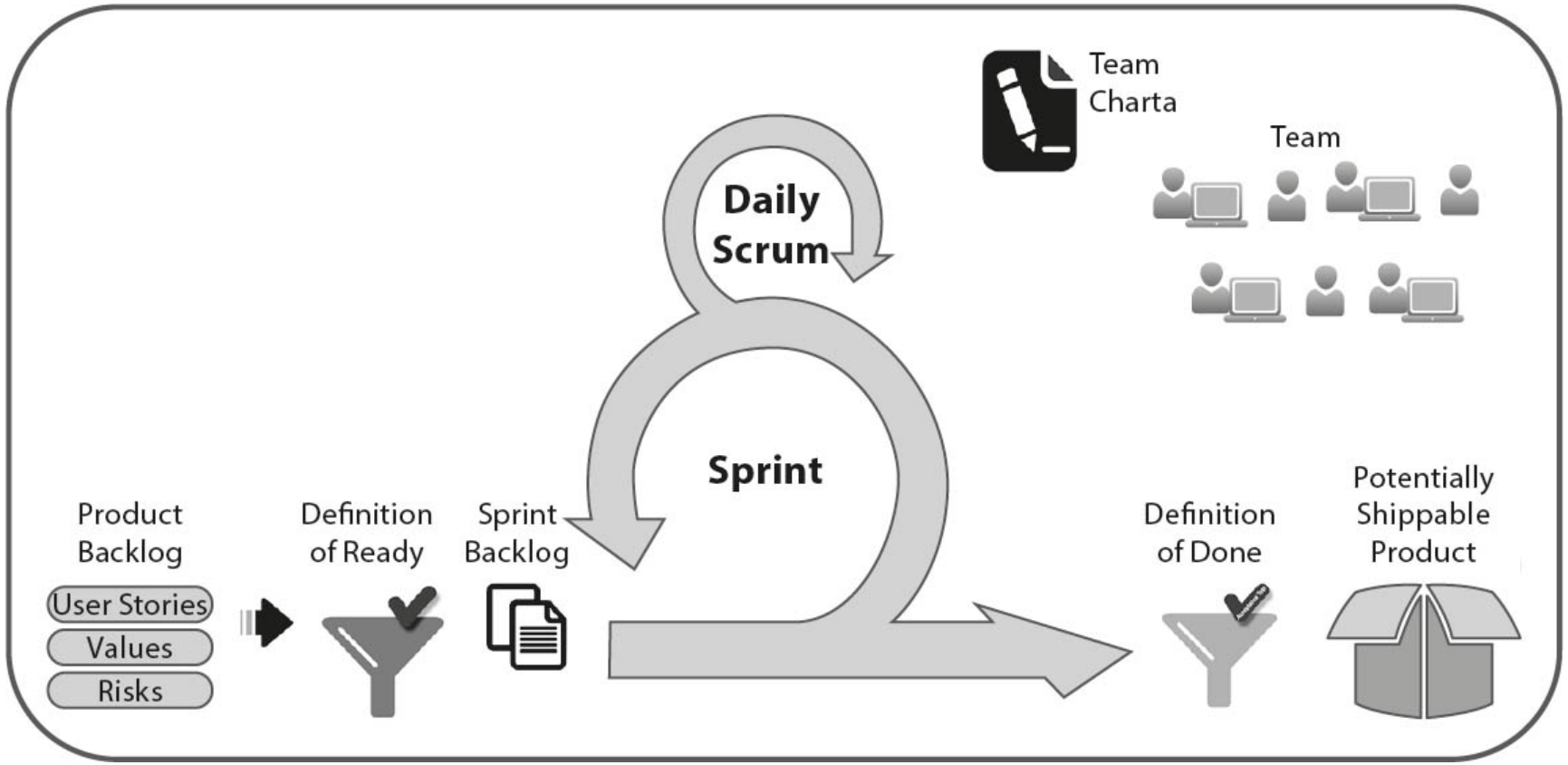
- Sürekli işbirliği ve iletişim gerektirir.
  - İletişimde boşluklar varsa veya ekip üyeleri dağılmışsa strateji daha az etkili olabilir.
- Değişimlere hızla uyum sağlayabilecek, bilgili ve deneyimli bir gruba ihtiyaç vardır.
  - Deneyimi olmayan ekipler bu modelin dinamik yapısını yönetmekte zorlanabilir.
- Karmaşıklık, özellikle büyük projelerde, artışların veya yinelemelerin yönetilmesiyle ortaya çıkarılabilir.
  - Entegrasyon ve senkronizasyonu garanti etmek için iyi bir proje yönetimine ihtiyaç vardır.
- Evrimsel modeller sürekli test, kullanıcı geri bildirimi ve prototip oluşturmayı gerektirdiğinden, daha yüksek bir başlangıç maliyeti içer.
  - Bu sınırlı finansmanı olan projelerde sorun olabilir.

Çevik Yazılım Geliştirme Yöntemi

- **Iterative and incremental** yaklaşımı gösterir.
  - Böylece ürün artırımsal olarak küçük parçalara bölünerek iterasyonları gerçekleştirir.
- Her bir iterasyon **Planning, Requirement Analysis, Design, Coding, Unit Testing, Acceptance Testing** adımlarını gerçekleştirir.
- Çevik yaklaşıma aynı zamanda gereksinimleri belirlemek ve gerçekleştirmek üzere müşteri ile sürekli iletişimde bulunulan (continuous iteraction) yöntem adı da verilir.



Agile Software Development Methods



- ❑ Extreme Programming (XP) Beck 2004
- ❑ Kanban
- ❑ Scrum

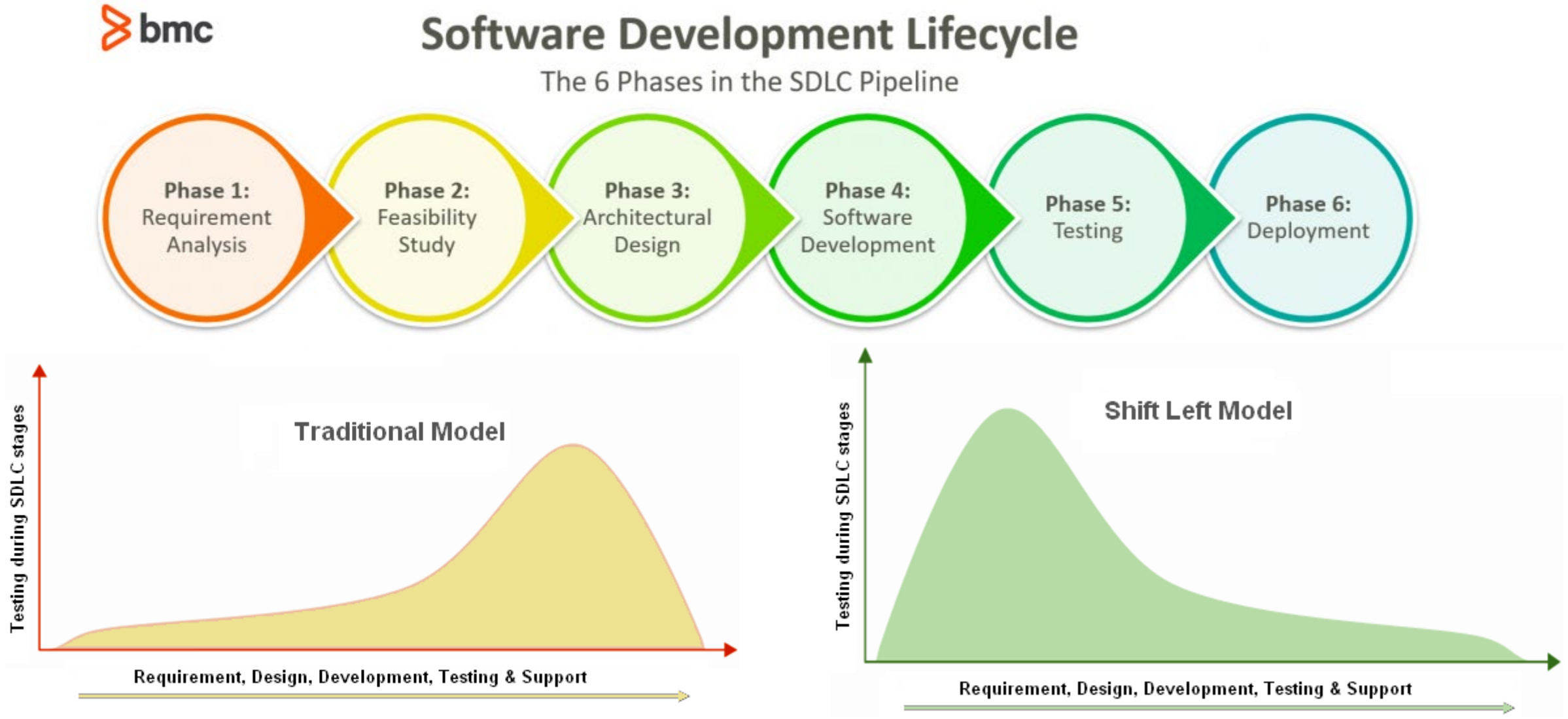
Her biri iterative /incremental geliştirme modelidir

Yeni Yazılım Geliştirme Metodolojilerinde Test

- Test, kısa süreli döngülere uygun olmalıdır.
  - Her bir bileşen, her yeni artış için anında tekrarlanabilen yeniden kullanılabilir test senaryoları gerektirir.
    - ✓Aksi durumda «increment» gerçekleştikçe sistem güvenilirliğinin düşüşe geçme riski vardır.
  - Her artış(increment) , herhangi bir ek işlevi kapsayan yeni test senaryoları gerektirir
    - ✓Bu da ürünün her sürümünde yürütülmesi gereken test senaryolarının sayısının zamanla arttığı anlamına gelir.
- Test otomasyonu, test çevik geliştirmeye uyarlanırken önemli bir araçtır.
- Çevik Yöntemler sürekli test (continuous test) gerektirir.
  - Bu da kaliteli ürün geliştirilmesi (high quality) demektir.

Shift Left Testing

- Testin döngü içerisinde geliştirmenin en erken zamanında başlamasıdır.
- Teste erken başlanması «bug» ların daha az maliyetle fazla zaman harcamadan bulunup ortaya çıkarılmasıdır (fix the defects)
  - Böylece CI (Continuous Integration) gerçekleşir. Test otomasyonu gerçekleştirilebilir.
- Yazılım Geliştirme Yaşam Döngüsü (SDLC) Continuous Testing ve CI/CD pipeline üzerinde gerçekleşir.
- Bu da DevOps yaklaşımını verecektir.



A defect that is removed after the product has gone into product will cost around 100 times more than one that is identified and removed during the requirements phase.

https://www.bmc.com/blogs/what-is-shift-left-shift-left-testing-explained