**BLM 426 YAZILIM MÜHENDİSLİĞİ**

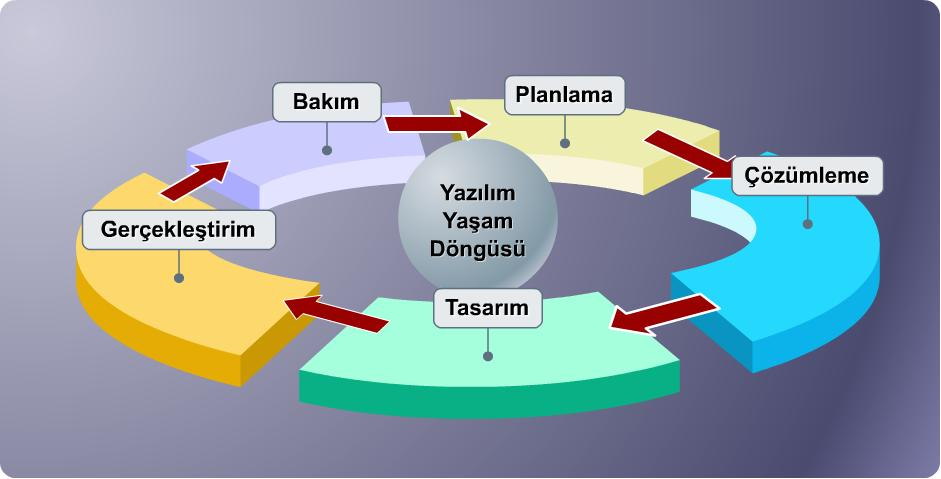
**BAHAR 2016**

**Yrd. Doç. Dr. Nesrin AYDIN ATASOY**

**2. HAFTA: YAZILIM SÜREÇ VE ÜRÜN TİPLERİ**

Yazılım geliştirmeyi sistematik hale getirmeyi hedefleyen çeşitli süreç modelleri ve yeni teknolojiler bulunmaya çalışılmaktadır. Model, yazılım geliştirme faaliyetinin nasıl yapılacağına, genel geliştirme düzeninin nasıl olacağına dair bir rehber niteliği taşır. Bu modellerin temel hedefi; proje başarısı için, yazılım geliştirme yaşam döngüsü (“software development life cycle”) boyunca izlenmesi önerilen mühendislik süreçlerini (önceden belirlenmiş adımlardan oluşan iş akışı) tanımlamaktır.

* **Yazılım Geliştirme Yaşam Döngüsü** (Software Development Life Cycle)
  + Herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar olarak tanımlanır.
  + Yazılım işlevleri ve ihtiyaçlar sürekli değiştiği ve geliştiği için bir döngü biçiminde düşünülür.
  + Yazılım yaşam döngüleri tek yönlü, doğrusal olarak düşünülmemelidir.
* **Yazılım Yaşam Döngüsü Temel Adımları**
* **Planlama:** Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.
* **Çözümleme:** Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Var olan işler incelenir, temel sorunlar ortaya çıkarılır.
* **Tasarım:** Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.
  + - Mantıksal tasarım; önerilen sistemin yapısı anlatılır,
    - Fiziksel tasarım; yazılımı içeren bileşenler ve bunların ayrıntıları.
* **Gerçekleştirim:** Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.
* **Bakım:** Teslimden sonra hata giderme ve yeni eklentiler yapma aşamasıdır.



**Şekil 1.** Yazılım yaşam döngüsü temel adımları.

* **Yazılım Yaşam Döngüsü Verileri**

Yazılım geliştirme standartları, yaşam döngüsü verilerinin hazırlanmasını gerektirir. Bazı standartlar verilerin hazırlanma şekli ve biçimini zorunlu tutarken, bazıları da yalnızca içeriği tanımlar, belge biçimini serbest bırakır.

Yaşam döngüsü verileri güncellenebilmeli, okunmalı, silinmeli, arşivlenmeli ve gerekliyse sahiplik haklarıyla birlikte devredilebilmelidir. Bir yaşam döngüsü verilerinin hazırlanmasının ana amacı şunlardır:

* Yazılım ürününün yaşam döngüsü boyunca gerekli olan bilgilerini tanımlamak ve kaydetmek,
* Yazılım ürününün kullanılabilirliğine yardımcı olmak,
* Yaşam döngüsü süreçlerini tanımlamak ve denetlemek,
* Veri değişikliğinin geçmişini tutmak,

Projelerin yaşam döngüsü verileri genellikle belgelerde saklanır. Böylece bu belgeler bir projeyi sağlıklı bir şekilde gerçekleştirebilmek ve ürün bakımını yapabilmek için çok önemlidir. Çünkü belgeler bilgilerin toplandığı düzenli ortamlardır. Bilgilerin sahip olması gereken özellikleri şöyledir:

* Çelişkisizlik
* Tamlık
* Doğrulanabilirlik
* Tutarlılık
* Değiştirilebilirlik
* İzlenebilirlik
* Sergilenebilirlik
* Gizlilik
* Korunmuşluk
* Hassaslık
* **Yazılım Süreç Modelleri**

Yazılım mühendisliği bir bilim dalı olmasına rağmen henüz diğer alanlar gibi anlaşılabilir değildir. Çünkü sabit temelleri yoktur. Hızlı bir şekilde yazılım geliştirmek için de büyük çalışmalar vardır. Yazılım teknolojilerinin gelişmesi ile var olan model ve metodolojiler de gelişmekte ve yeni modeller ortaya çıkmaktadır. Modellerin ortaya çıkmasında, ilgili dönemin donanım ve yazılım teknolojileri ile sektör ihtiyaçları önemli rol oynamıştır.

Uygun yazılım geliştirme modelleri kullanılması, yazılımın daha emniyetli, doğru, anlaşılabilir,  test edilebilir ve bakım yapılabilir olarak geliştirilmesinde çok önemli rol oynar.

* **Süreç nedir?**

Belirli bir hedef için gerçekleştirilen adımlar zinciridir. [IEEE]

* **Yazılım süreci nedir?**
* Yazılımı ve ilişkili ürünlerini geliştirmek ve idame ettirmek için kullanılan etkinlikler, yöntemler, pratikler ve dönüşümlerdir.
* Yazılım geliştirme ve idame amacı güden etkinlikler setidir.
* **Yazılım süreç modeli nedir?**

Bir yazılım sürecinin belirli bir bakış açısıyla gösterilmiş, basitleştirilmiş temsilidir. Örnek bakış açıları:

**İş-akışı:** Etkinlikler nasıl sıralı?

**Veri-akış:** Bilgiler nasıl sıralı?

**Rol-hareket:** Kim ne yapıyor?

**Yazılım Süreç Modeli Çeşitleri;**

1) Klasik Modeli

2) V Modeli

3) Prototipleme/Örnekleme

4) Spiral model

4.1)Evrimsel geliştirme

4.2)Evrimsel prototipleme

4.3)Artımlı geliştirme

4.4)Araştırmaya dayalı

5) Gelişigüzel geliştirme

6) Yeni teknikler

6.1)Özneye yönelik geliştirme

6.2)Bileşen tabanlı Geliştirme

6.3)Özelliğe yönelik programlama

6.4)Uç programlama

7) Tekniklerin birleştirilmesi

Şimdi bu yöntemleri inceleyelim;

## Klasik Model (Şelale Modeli)

Sistematik olarak ilerleyerek ardışık bir yaklaşımla yazılım geliştirmesini sağlanır. Gereksinimler ve isterler belirlenir. Buna göre tasarım yapılır. En eski ve en yaygın geliştirme modelidir. Yazılımın gelişimi, doğrusaldır. Özellikle iyi tanımlı isterleri kesinleşmiş fazla zaman istemeyen projeler için uygundur.

Gereklilik Tanımlaması



Sistem ve yazılım tasarımı Tanımlaması

Faaliyet ve bakım

Bütünleştirme ve birim testi

Uygulama ve birim testi

**Şekil 2.** Klasik model etkinlik adımları.

* Yaşam döngüsü temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir.
* Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.
* İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
* Yazılım tanımlamada belirsizlik yok (ya da az) ise ve yazılım üretimi çok zaman almayacak ise uygun bir süreç modelidir.

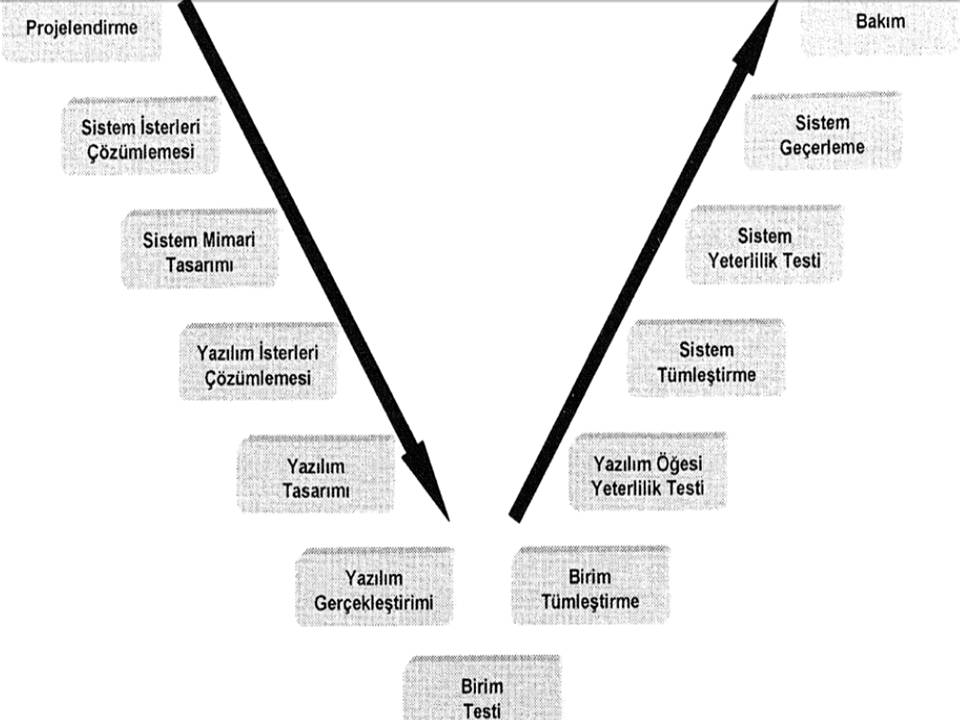
Bu model en eski ve en yaygın yazılım geliştirme tekniğidir, bu nedenle son yıllarda çeşitli eleştirilere uğramıştır. Bunlardan bazıları şöyledir:

1. Gerçek yaşamdaki projeler genelde yineleme gerektirir. Bu modelde her adım en az bir kere işlediği için maliyet ve teslim zamanı artar.
2. Müşteri tüm isterlerini bir defada açıklayamayacağı için projenin başında belirsizlikler ortaya çıkar.
3. Müşterinin projenin çalışan bir sürümünü görebilmesi ilerleyen basamaklarda olacağı için müşteri hayal kırıklığına uğrayabilir ve proje başarısızlığa gidebilir.
4. Geliştirici personel genelde kod yazma eğilimlidir. Bu nedenle gereklilik tanımlanması ve tasarım gibi süreçler personele sıkıcı gelebilir.

Müşterinin ürünü tasarımı bittiğinde görmesi yerine geliştirme sürecinin her aşamasında onaylayarak sürece katılması projeyi olumlu etkiler. Klasik model diğer teknikler için bir kalıp oluşturur. Özellikle iyi tanımlı, isterleri kesinleşmiş ve fazla zaman alması beklenmeyen projeler için uygun bir yöntemdir.

## V Modeli

Klasik modeldeki test işlemlerinin ne zaman yapılacağını ön plana çıkarır ve testler sırasında bulunan hataların düzeltilmesi için hangi düzeye dönülmesi gerektiğini gösterir. Sol kanat üretim etkinlikleri, sağ kanat test etkinlikleri ile ilgilidir.



**Şekil 3.** V model etkinlik adımları.

Örneğin, Sistem geçerleme sırasında oluşan bir sorunu çözebilmek için sistem isterleri çözümlenmesinin tekrarlanması gerekmektedir.

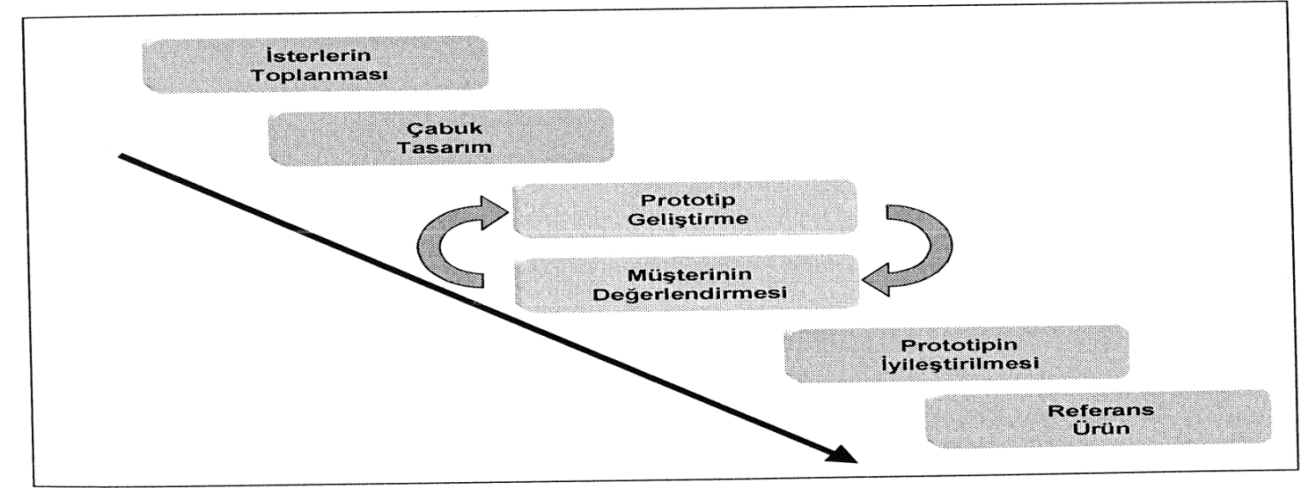
İsterlerin iyi tanımlandığı, belirsizliklerin az olduğu projeler için uygundur.

## Prototipleme/Örnekleme

Yazılım geliştirme sürecinde belirsizlik varsa bu yöntem idealdir. Geliştirilecek olan yazılımın bir çalışan modeli yapılarak şüpheli görülen noktalarda değerlendirme yapılır. Diğerlerindeki gibi işe yazılım isterlerinin toplanmasıyla başlanır. Üzerinde daha fazla düşünülmesi gereken noktalar belirlenir. Ortaya çıkan prototipi ilk ürün olarak kabul edip sistem haline dönüştürülebilir ya da tamamen vazgeçilip yeni bir prototip geliştirilebilir.

Prototip ürün bittiğinde müşteri yazılım ürününün çok çabuk bittiğini düşünebilir. Fakat prototip olarak geliştirilen bu üründe nitelik ve bakım aşaması yoktur.

Yazılım geliştiricinin seçtiği dil, işletim sistemi vs. gibi durumlar ürün geliştirirken yeterli olmayabilir. Asıl sistemide bu deneyimlerle gerçekleştirirse bu seçimler sistemin zayıf noktaları olarak kalır.



**Şekil 4.** Prototipleme model etkinlik adımları.

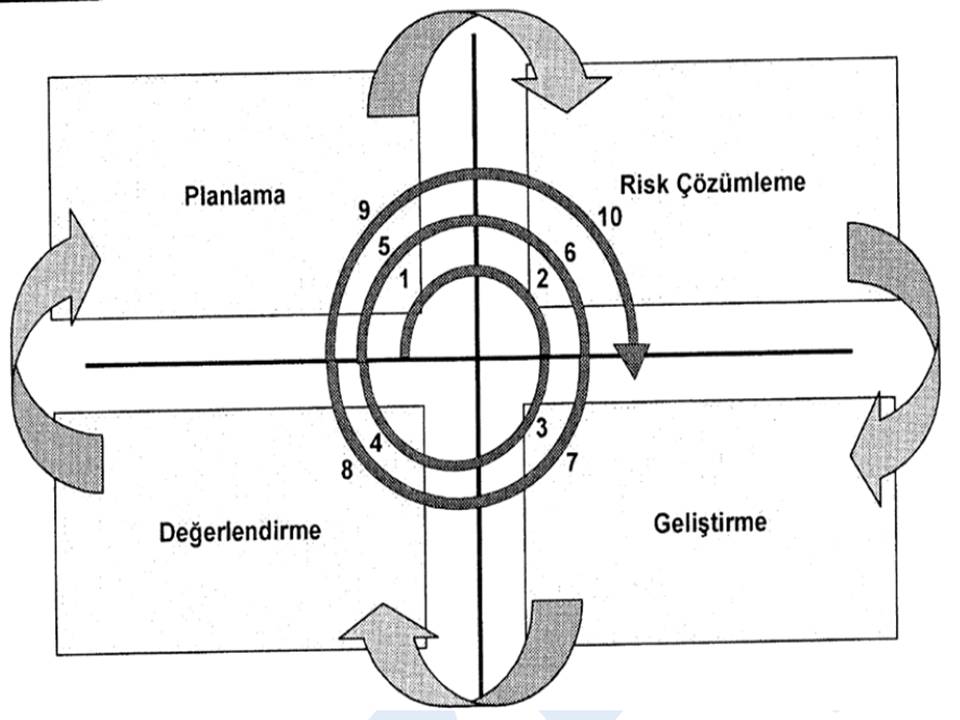
## Spiral model

Bu model klasik model ve prototipleme modellerinin iyi yönlerinin birleştirilmesiyle oluşturulmuştur.

Bu modelde kullanıcının kesin olarak gereksinimlerinin bir kısmı belirlenir ve bunlardan bir kısım ister tanımlanır. Bunların gerçekleştirimi yapıldıktan sonra ürünün testi yapılarak müşteriye teslim edilir. Art arda tekrarlanan 4 aşamadan oluşur.

* Planlama aşaması; amaçların, olası seçeneklerin ve kısıtlamaların değerlendirildiği aşama
* Risk çözümlemesi; risklerin tanımlandığı ve çözüm yöntemlerinin irdelendiği aşama
* Mühendislik; ürünün geliştirildiği aşama
* Değerlendirme, ürünün müşteriyle birlikte değerlendirildiği aşama.

Bu aşamalar ürününün tamamlanmasına kadar geçen sürede tekrarlandığı için spiral halinde gösterilmektedir. Her bir çeyrek bir diğerinin girdisini oluşturarak devam eder.

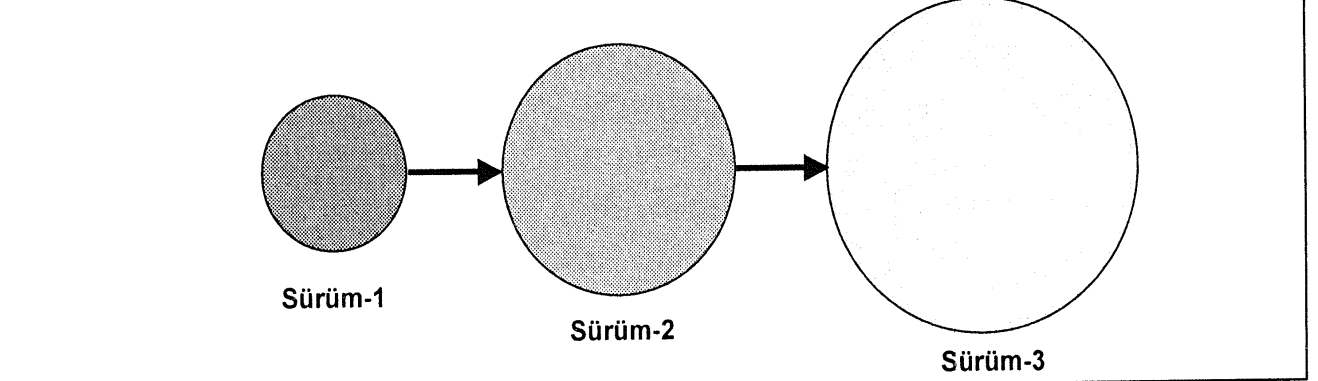


**Şekil 4.** Spiral model.

Bu modelin 3. Çeyreğinde mutlaka klasik model ya da prototipleme gibi bir model kullanılmalıdır. Spiral merkezinden uzaklaştıkça bu aşamadaki geliştirme işleri daha da artar. Klasik çevrim geliştirme için, prototipleme ise riskleri azaltmak için kullanılır. Klasik çevrimi esas alaran bazı geliştirme yöntemleri şöyledir:

1. **Evrimsel geliştirme**

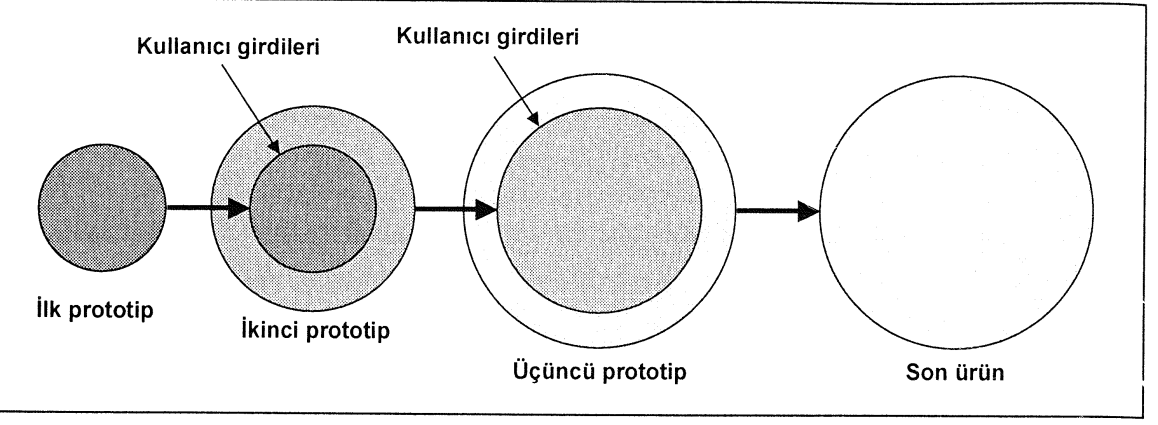
Evrimler halinde ürün ortaya çıkması hedefler. Başarı ilk evrimde ortaya çıkan pilot ürüne bağlıdır. Her evrimde geliştirilen ürünler uygulama alanında tam işlevselliğe sahiptir. Ortaya çıkan her ürün teslim edilerek kullanıma sunulur. Her evrimde, gelen geri bildirimlere, deneyimlere ve yeni gereksinimlere göre sistemin kapsamı, işlevleri ve yetenekleri geliştirilir.



**Şekil 5.** Evrimsel geliştirme.

1. **Evrimsel prototipleme**

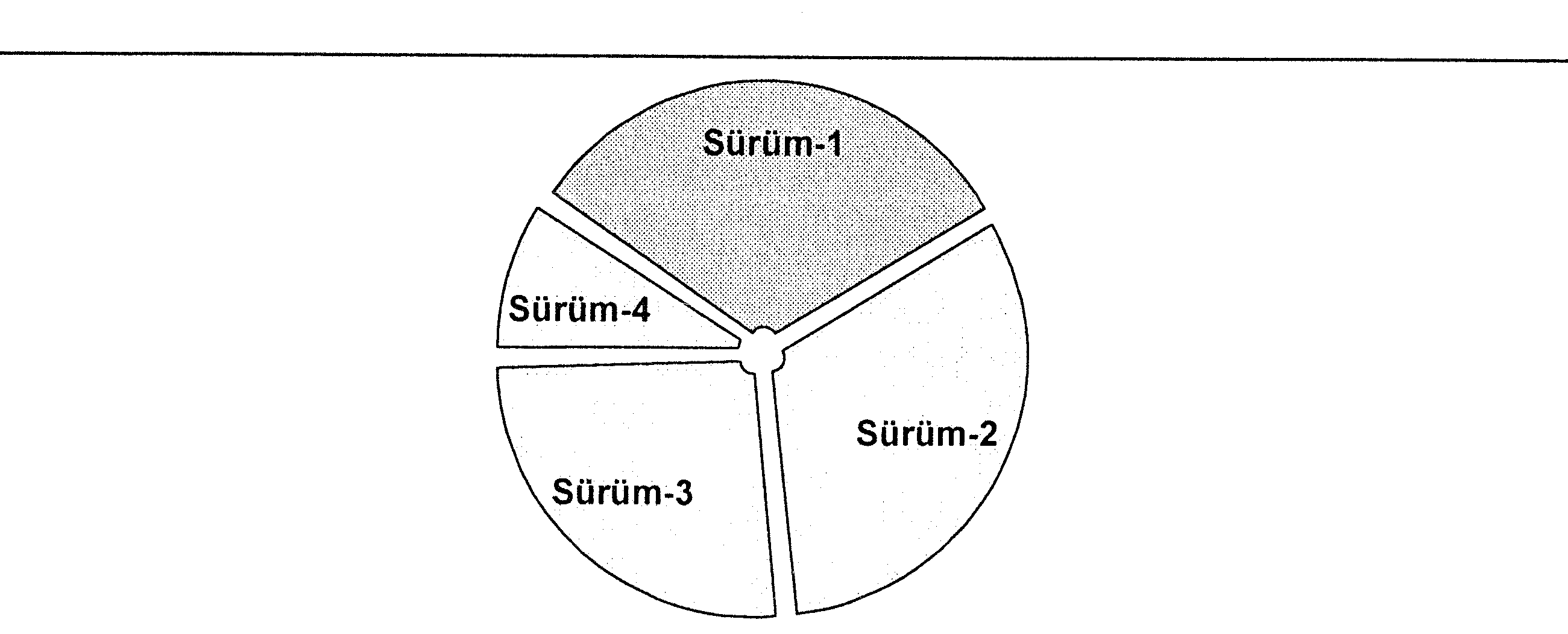
Evrimsel geliştirmeden farklı olarak, her evrimde ortaya prototip yani yeni bir ürün çıkar. Ön ürünün uygulama alanında denenmesinden sonra kullanıcı geri bildirimleri alınarak başka bir ön ürün oluşturulur ve denemeye sunulur. Ara aşamalarda müşteriye deneme ürün verilir. Bu şekilde geliştirmeye devam edilerek ürün son haline getirilerek kullanıma sunulur.



**Şekil 6.** Evrimsel prototipleme.

1. **Artırımlı geliştirme**

İsterlerin tamamı belirli olan bir ürünün sürümler şeklinde geliştirmesine dayanır. Önce en temel isterlere göre çekirdek yapıda bir ürün geliştirilir. Bu ürün sistemin temel işlevlerini yerine getirebilecek durumdadır ve bazı isterler eksiktir. Sonra her bir yeni sürümde isterlerin bir kısmı daha eklenerek çekirdek ürün geliştirilmiş olur. Bu model, evrimsel geliştirmeye benzemekle beraber en büyük farkı, sürümlerin tam işlevsellik içermemesidir.



**Şekil 7.** Artırımlı geliştirme modeli.

1. **Araştırmaya dayalı geliştirme**

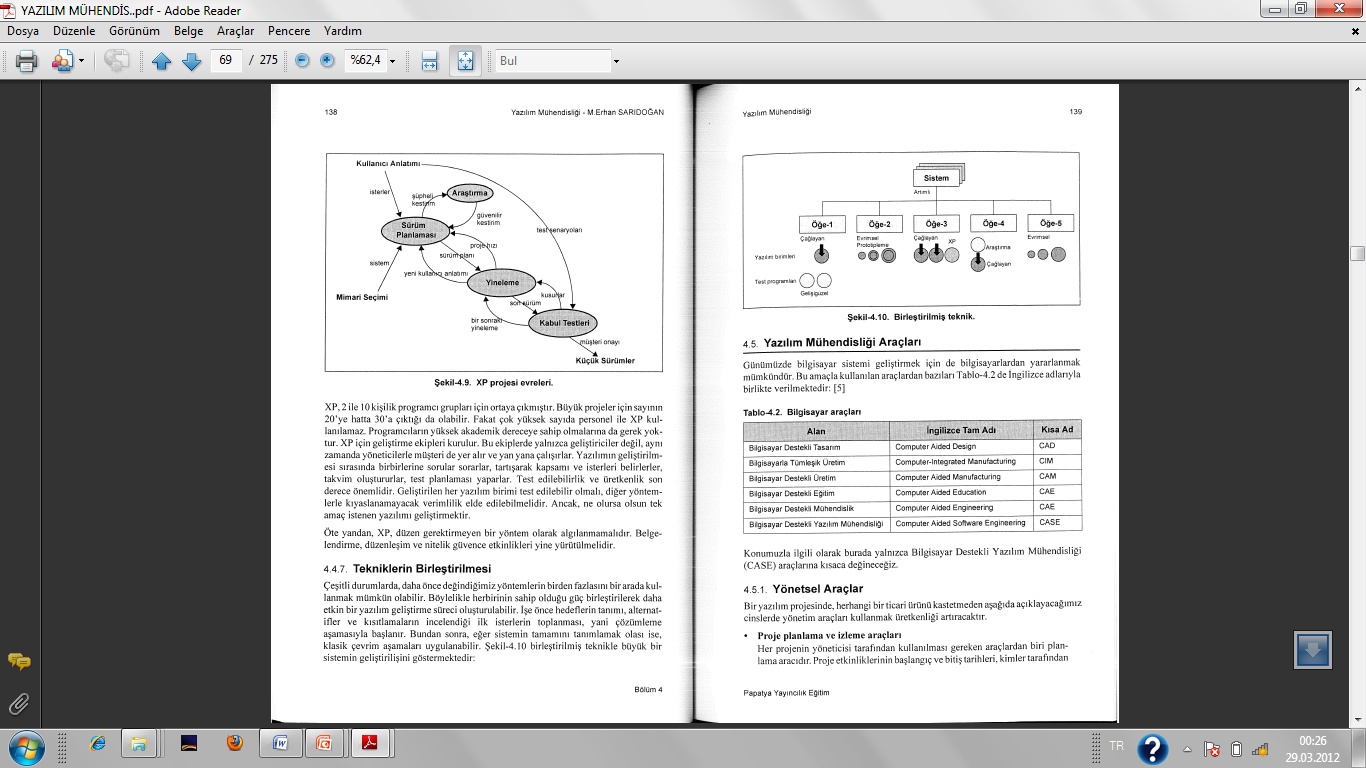
Tam bir belirsizlikle başlar. Araştırmaya dayalıdır. Sonuçlar hakkında kesin bir şey söylenemez. Bu model özellikle kritik sistemlerin bazı kısımlarında ilk kez kullanılacak bir teknolojinin uygulaması ve algoritma geliştirilmesinde kullanılır.

1. **Gelişigüzel geliştirme**

Ürünün sadece çalışması yeterlidir. Gelişigüzel bir şekilde kod üretilir. Herhangi bir model ya da yöntem yoktur. Maalesef pek çok yazılım geliştirici bu yöntemi tercih etmektedir ama bunun yazılım mühendisliğinde yeri yoktur.

1. **Tekniklerin birleştirilmesi**

Bazı durumlarda yukarıda saydığımız teknikler bir arada kullanılabilir. Böylelikle yazılım daha da güçlenir. İsterlerin toplamı, hedeflerin tanımı ile başlanır. Daha sonra diğer yöntemler uygulanabilir.



**Şekil 7.** Birleştirilmiş teknik.

**KAYNAKLAR**

1. Yazılım Mühendisliği Ders Notları; Yrd.Doç.Dr. Buket Doğan.
2. Yazılım Mühendisliği Ders Notu; Cengiz GÖK
3. Yazılım Mühendisliği; M. Erhan Sarıdoğan