

YAZILIM PROJESİ GELİŞTİRME DERS NOTLARI

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM PROJELERİNİN BUGÜNKÜ DURUMU VE İSTATİSTİKLER

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM PROJESİ GELİŞTİRME

- **Yazılım**; bilgi teknolojileri (BT) projelerinin en önemli bileşeni olup, insanların belirli konularda ihtiyaçlarını karşılamak ve istedikleri işlevleri (fonksiyonları) yerine getirebilmek amacıyla geliştirilen; **programlar, veriler ve belgeler** topluluğudur.
- BT, işletme maliyetlerinin en önemli kalemi de personel maliyetidir. **Personel işgücünü en verimli şekilde kullanmak BT firmalarının en önemli hedefidir.**
- BT operasyonlarında, teknik destek dışında yazılım geliştirme önemli yer tutmaktadır. Bu sebeple, yazılım geliştirme süreçlerinin ve aşamalarının standartlara uygun ve süreçlerin eksiksiz / doğru bir şekilde uygulanması **hem personelin işgücünü verimli kullanmak hem de maliyetleri düşürmek anlamında çok önemlidir.**

YAZILIM PROJESİ GELİŞTİRME

- Yazılım hataları, yazılım yaşam döngüsünde çok önemli yer tutan unsurlardan biridir.
- Teorik olarak bir yazılım üretim aşamasında tüm ayrıntıları ile test edilebilir görünmesine rağmen uygulamada özellikle **orta ve büyük ölçekli projelerde** bu mümkün olamamaktadır.
- İşletmeye alınan ve üretimi tamamlanmış her yazılım projesi; %100 hatasız çalışıyor anlamına gelmez.
- Her zaman bir hatanın ortaya çıkma olasılığı vardır.
- Yazılım proje hatalarının en önemli nedenlerinden birisi; **geliştirme öncesi analiz, tasarım ve planlama aşamalarından kaynaklı eksiklikler ve yanlışlıklarından** kaynaklanmasıdır.

YAZILIM PROJESİ GELİŞTİRME

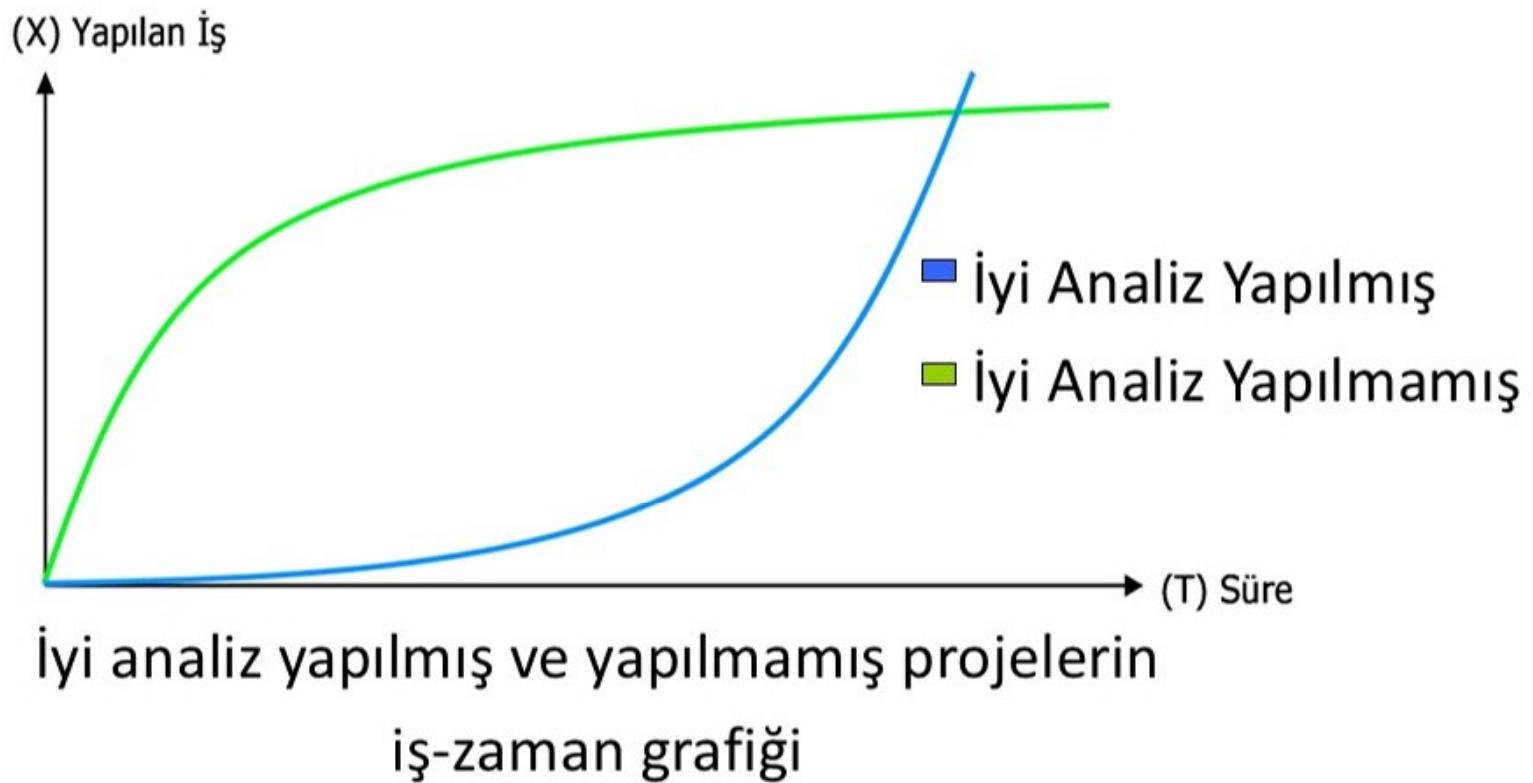
- Aşağıdaki grafikte; planlaması, analizi ve tasarımları iyi yapılmış (standartlara ve kurallara uyulmuş) projeler ile yapılmamış projeler arasındaki fark gösterilmiştir.
- Grafikten de görüleceği gibi; planlaması, analizi ve tasarımları iyi yapılmamış projelerde başlangıçta iş yapılmış/ ilerleme kaydedilmiş görünmesine rağmen sonrasında yapılan iş devam ekte ve zaman içerisinde tamamlanamamaktadır.
- Analizi, tasarımları ve planlamaları iyi yapılan yazılım projelerinde ise başlangıçta yapılan iş / ilerleme görünmemesine rağmen bu tip projeler daha sonra kısa sürede tamamlanmakta ve hata çıkma olasılığı da çok daha düşük olmaktadır.

YAZILIM PROJESİ GELİŞTİRME

Yazılım projelerinde; hataları minimuma indirmenin en önemli adımı; planlama, analiz ve tasarım aşamalarının kurallara / standartlara uygun şekilde geçilmesidir.

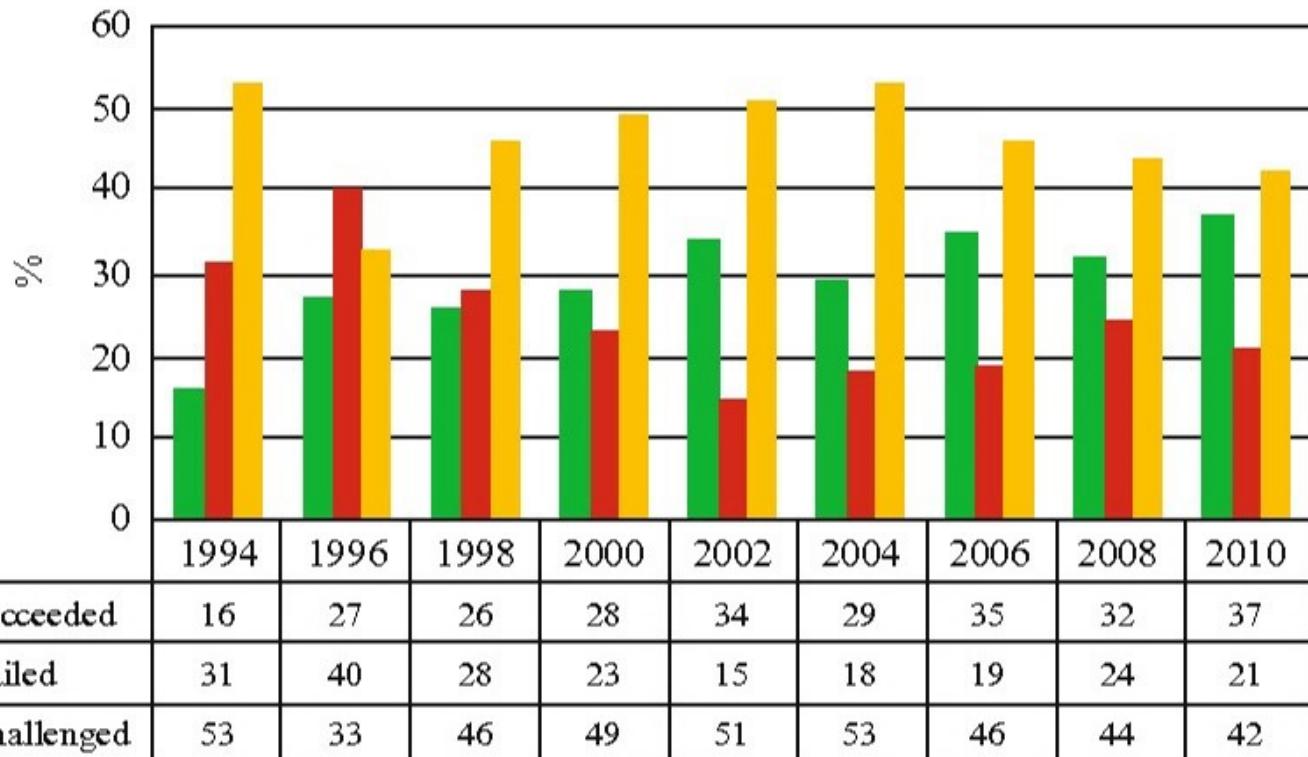
SÜREÇLER:

- Planlama
- Analiz
- Tasarım
- Geliştirme
- Test (IT ve Kullanıcı)
- Canlı & Production



YAZILIM PROJESİ GELİŞTİRME

FARKLI ÖLÇEKLERDEKİ TÜM PROJELER İÇİN BAŞARI ORANLARI



KAYNAK: STANDISH

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM PROJESİ GELİŞTİRME

YAZILIM PROJELERİ BAŞARI ORANLARI

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

KAYNAK: STANDISH - 2020

YAZILIM PROJESİ GELİŞTİRME

PROJE BÜYÜKLÜKLERİNE GÖRE BAŞARI ORANLARI

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

KAYNAK: STANDISH - 2020

YAZILIM PROJESİ GELİŞTİRME

ÇEVİK VE KLASİK YÖNTEMLERİN KARŞILAŞTIRILMASI

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

KAYNAK: STANDISH - 2020

YAZILIM PROJESİ GELİŞTİRME

Yukarıdaki tablolar incelendiğinde projelerdeki başarı oranının yıllar içerisinde biraz arttığı görülmekte olup, yine de yeterli seviyede olmadığı aşikardır.

Başarı seviyesindeki yetersiz artışın nedenlerinden bazıları aşağıdadır:

- Yazılım mühendisliğindeki ilerlemeler, (yeni metodolojiler / yeni yaklaşımlar, araçlar / tool'lar, vs)
- Standartların kullanılmaya başlanması (kısmen de olsa)
- Yazılım kalite kontrolünün öneminin anlaşılması,
- Daha Detaylı Testler, Test Senaryoları ve Test Case lerin oluşturulması, Test Tool'larının kullanılmaya başlaması, Test Personelinin Bilinçlendirilmesi, Ödül Sistemi vs.
- Servis ve Müşteri Odaklı Yaklaşım,
- Uzmanlık alanlarının küçülmesi, daraltılması (developer, tester, designer, analyst vs)

YAZILIM PROJELERİNDE HATA DAĞILIMLARI

Dr. Öğretim Üyesi Yüksel BAL

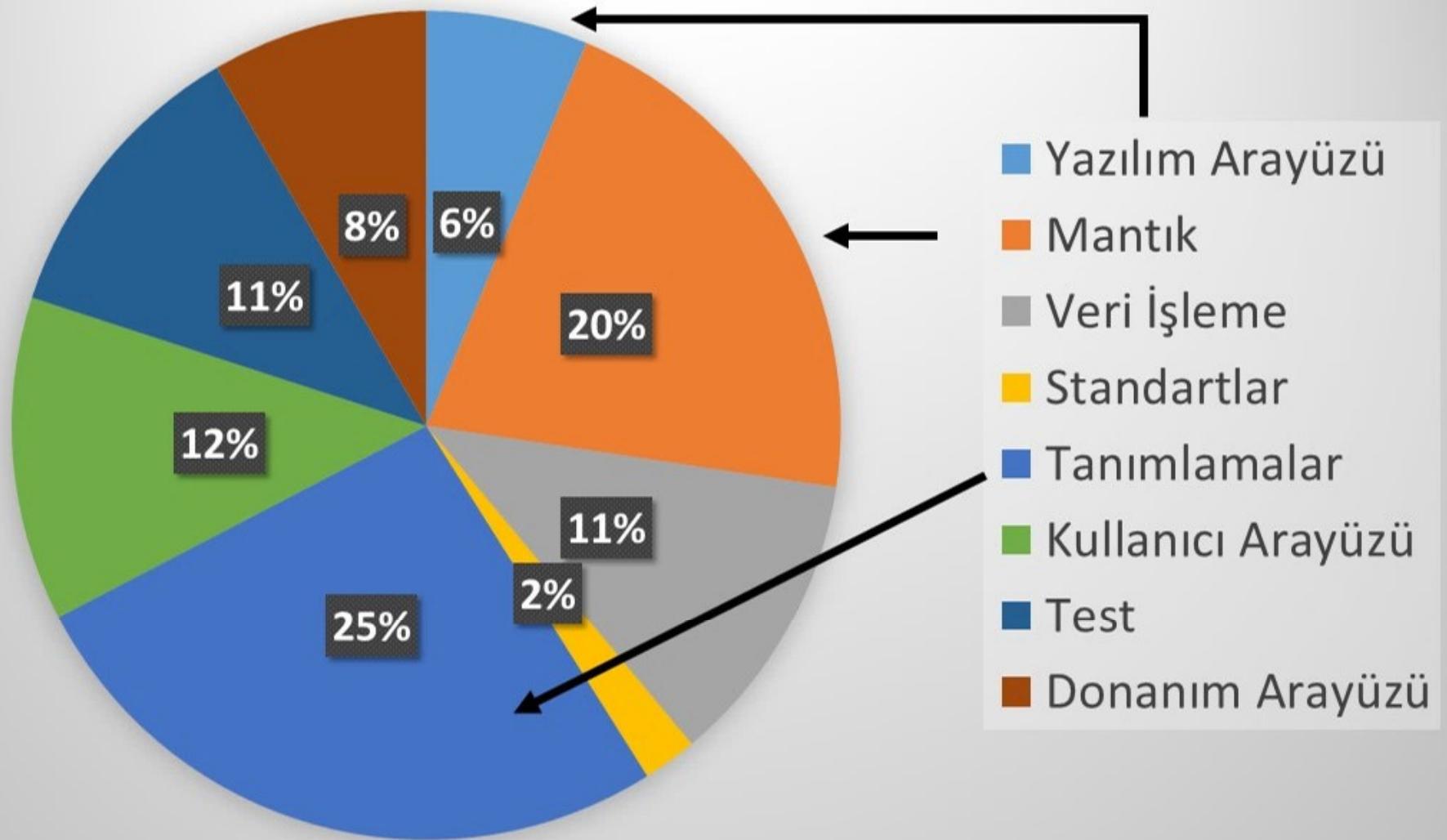
YAZILIM PROJESİ GELİŞTİRME

Yazılım Projelerinde Süreçlere Göre Hata Dağılımları

Yazılım projesi geliştirmede, projenin başından sonuna kadar olan tüm süreçler dikkate alındığında hataların nerelerde yapıldığına dair **hata dağılım grafiği** aşağıda verilmiştir.

Hata dağılımları konusunda farklı bakış açılarından kaynaklanan farklı değerlendirmeler ve yorumlamalar yapılmasına rağmen, genelde hata dağılımları birbirlerine yakın verilmektedir.

Yazılım Üretim Hatalarının Dağılımı



YAZILIM PROJESİ GELİŞTİRME

➤ Diğer taraftan, tüm değerlendirmelerde ortaya çıkan hatalarda, **hataların oluşmasında öne çıkan 3 konu** dikkat çekmektedir:

- Projelerin daha başında **tanımlama temel sürecindeki eksiklikler ve hatalar** (Gereksinim analizi, sistem analizi, tasarım, planlama aşamalarında)
- Projenin başından sonuna kadar her aşamada **belgelendirmenin yapılmaması veya yetersiz** oluşu,
- Test aşamasında, özellikle orta ve büyük ölçekli projelerde zaman kısıtlamasından dolayı çok fazla ‘test case’ oluşundan kaynaklı **testlerin gerekli olan veya tüm fonksiyonları ile test edilememesinden kaynaklı** hataların gözden kaçmasıdır.

YAZILIM PROJESİ GELİŞTİRME

➤ Tüm bu sebeplerden dolayı son zamanlarda hata oranlarını düşürebilmek için yazılım geliştirme süreçlerinde;

- ✓ Farklı metodoloji ve proje yönetim yaklaşımları benimsenmeye başlanmış,
- ✓ Yeni yaklaşımların hepsinde müşteri ekibin içerisinde dahil edilmiş,
- ✓ Özellikle orta ve büyük ölçekli yazılım projelerinde; prototip geliştirmeli, sürüm artımlı geliştirme modellerine doğru bir geçiş başlamıştır.

Tüm bunların sonucunda; müşterinin de ekibin içerisinde olduğu özellikle orta ve büyük ölçekli projelerde fazlandırmış / aşamalı / spiral döngülü / sürümlü / prototip artımlı ilerlemeleri benimseyen yazılım proje geliştirmeleri artmaya başlamıştır.

YAZILIM PROJESİ GELİŞTİRME

Aşağıda, yazılım hata düzeltme maliyetleri tablosundan görüleceği gibi, yazılım geliştirme süreçlerinde yapılan hatalarda; hatalar ne kadar **erken anlaşılırsa maliyeti o kadar az** olmaktadır.

- Bilgi teknolojileri projelerinde en büyük maliyet insan kaynakları maliyetleridir. Yani; analist, tasarımcı, programcı, tester gibi insan kaynaklarının **iş gücü (zaman) maliyetleridir.**
- Bu sebepten dolayı; yazılım geliştirmede aşamalar ilerledikçe ortaya çıkması muhtemel hataların maliyeti de **ilerleyen her aşamada gittikçe artmaktadır.**

YAZILIM PROJESİ GELİŞTİRME

- Hatalar, fark edilmeden proje canlı ortama alınırsa hata maliyet oranı: %100 olmaktadır. Yani bu durum, bu projeyi yeniden yapıyormuşuz gibi **iş gücü** ve **zaman** kaybına neden olacak bu da **maddi anlamda da maliyet** getirecektir.

YAZILIM PROJESİ GELİŞTİRME

SÜREÇLERE GÖRE YAZILIM HATA DÜZELTME MALİYETLERİ



İlerleme Yönü

Süreç	Hata Maliyet Oranı (%)
Çözümleme / Analiz	1
Tasarım	5
Kodlama	10
Test	25
Kabul Testi	50
İşletim (Canlı Ortam)	100

NOT: HATALARIN YAZILIM GELİŞTİRME SÜREÇLERİNE İLERİ YÖNDE YAYILMA ÖZELLİĞİ VARDIR.

Yazılım Projelerinin Sınıflandırılması

Dr. Öğretim Üyesi Yüksel BAL

Yazılımları İşlevlerine Göre Sınıflandırma

İŞLEV	KULLANIM ALANI
Hesaplama	Mühendislik Çözümleme
Veri İşleme	Bankacılık, Haberleşme Sistemleri
Süreç Temelli	Gömülü Sistemler
Kural Temelli	Robotik, Yapay Zeka
CAD	Sinyal İşleme

Zamana Dayalı Özelliklere Göre Sınıflandırma

Online: Kullanıcı tarafından talep yapıldığında belirli süre içerisinde cevap dönen sistem yazılımları, cevabın gecikmesi durumunda talep ortadan kalkana ya da “time out” süresine kadar beklerler.

Batch (Toplu): Belirlenmiş bir zamanda veya periyodik olarak topluca işlem yaptırılmak için çalışan/programlanan yazılımlardır. Manuel veya otomatik olarak çalıştırılabilirler.

Real Time (Gerçek Zamanlı): Çok kısa bir zaman dilimi (ns, μ s) içerisinde ulaşılabilen ve cevap döndürebilen ve kesintileri önlemek için alternatif erişim ve altyapıları olan yazılımlardır. (Uzay sanayi, savunma sanayi, ilaç sanayi vs yazılımları).

Yazılımları Boyuta Göre Sınıflandırma

YAZILIM SATIR SAYISI	KULLANIM ALANI
Küçük ($SS < 2000$)	PC Oyunları, Mühendislik Çözümleme
Orta ($2000 < SS < 100.000$)	CAD, BDE Yazılımları
Büyük ($100.000 < SS < 1 \text{ Milyon}$)	Uzay Mekiği
Çok Büyük ($SS > 1 \text{ Milyon}$)	Hava Tahmini, Kontrol Sistemleri, İşletim Sistemleri (OS), Savaş Uçakları ve savunma sanayi sistemleri

NOT: SS = Satır Sayısı,

CAD = Computer Aided Design (Bilgisayar Destekli Tasarım),

BDE = Bilgisayar Destekli Eğitim

YAZILIM YAŞAM DÖNGÜSÜ (THE SOFTWARE LIFE CYCLE)

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM YAŞAM DÖNGÜSÜ

Yazılım Geliştirme Metodolojileri

- **Yazılım**, yalnızca bir bilgisayar programı ya da belirli bir iş yapan paket program değil, bir mantık dahilinde insanlar tarafından oluşturulan ve kullanılan; program, veri ve belgeler topluluğudur.
- Bilgi teknolojilerinin temel amaçlarından birisi; yazılım geliştirmedeki karmaşıklığı gidererek sağlam, doğru, güvenilir ve isteğe uygun ürünleri daha düşük maliyetle ve ekonomik bir şekilde ortaya çıkarmaktır.
- Yazılım kendi başına bir *sistem* olmayıp, mutlaka daha büyük sistemin bir parçasıdır.
- Metodoloji; yazılım yaşam çevrimi boyunca kullanılacak süreç, belirtim, belgelendirme gibi yöntemler bütününu içeren ve süreçlerin ardışıl olarak nasıl sıralandığını belirten bir disiplindir.

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM YAŞAM DÖNGÜSÜ

Yazılım Geliştirme Metodolojileri

- Yapısal sistem geliştirme yöntemlerinin gelişmesiyle başlayan yapısal metodolojiler, veri akışına, veri yapısına ve nesneye yönelik metodolojiler halinde gelişmeye devam etmektedir.
- Uluslararası kuruluşlar tarafından çeşitli standartlar ve rehberler geliştirilmekte, ortak bir noktaya doğru ilerlemeye çalışılmaktadır.
- Bir metodolojide 2 temel unsur bulunmaktadır:
 - Belirtim yöntemleri
 - Süreç modelleri

YAZILIM YAŞAM DÖNGÜSÜ

BELİRTİM YÖNTEMLERİ

Bir çekirdek süreçte ilişkin işlevleri yerine getirmek amacıyla kullanılan yöntemler belirtim yöntemleri olarak anılmaktadır. Farklı işlevler için kullanılan belirtim yöntemleri üç sınıfa ayrılabilir.

1- Süreç akışı için kullanılan belirtim yöntemleri: Süreçler arası ilişkilerin ve iletişimimin gösterildiği yöntemlerdir.

- Veri akış şemaları,
- Yapısal şemalar,
- Nesne/sınıf şemaları

vb. belirtim yöntemlerine örnek olarak verilebilir.

YAZILIM YAŞAM DÖNGÜSÜ

2- Süreç tanımlama yöntemleri: Süreçlerin iç işleyişlerini göstermek için kullanılan yöntemlerdir. Farklı aşamalar için aynı ya da farklı süreç tanımlama yöntemi kullanılabilir.

- Düz metin,
 - Şablon,
 - Karar tabloları,
 - Karar ağaçları,
 - Algoritma anlatım dili
- bu tür yöntemlere örnek olarak verilebilir.

YAZILIM YAŞAM DÖNGÜSÜ

3- Veri tanımlama yöntemleri: Süreçler tarafından kullanılan verilerin tanımlanması amacıyla kullanılan yöntemlerdir.

- Nesne-iliski modeli,
- Veri sözlüğü,
- Veri tabanı tabloları,
- Programlama platformuna bağlı veri tanımlama yöntemleri

bu tür yöntemlere örnek olarak verilebilir.

YAZILIM YAŞAM DÖNGÜSÜ

SÜREÇ MODELLERİ:

Yazılım geliştirmede uygulanan süreç modelleri, yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, **hangi düzen ya da sıraya ve nasıl uygulanacağını tanımlar**. Süreçlerin içsel ayrıntıları ya da süreçler arası ilişkilerle ilgilenmez. Özetle, yazılım üretim işinin genel yapılması düzene iliskin rehberler olarak tanımlanabilirler.

1. Gelişigüzel Model
2. Barok Modeli
3. “V” Modeli

YAZILIM YAŞAM DÖNGÜSÜ

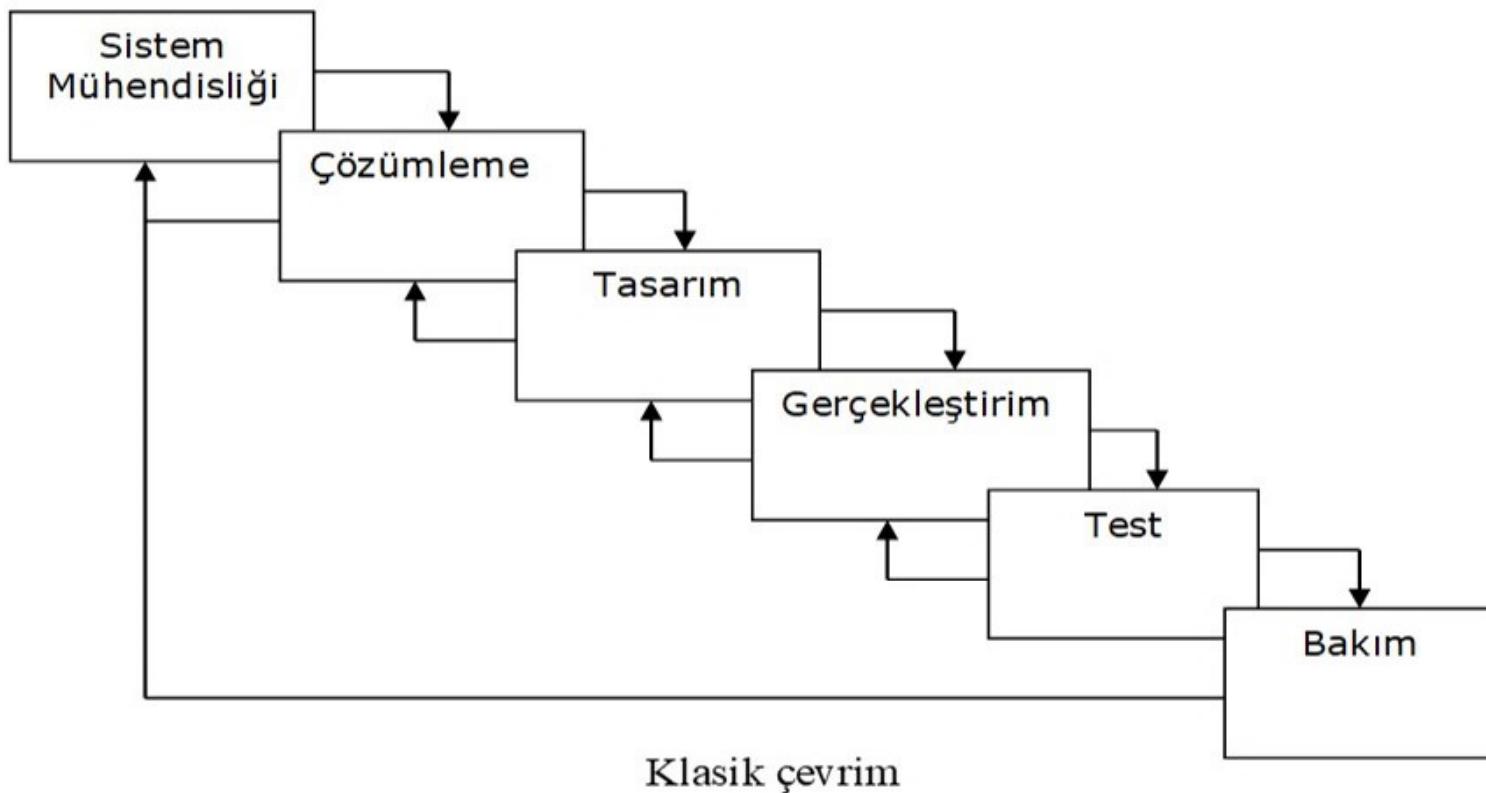
4. Klasik Çevrim

- 1970'li yılların ortalarında, yapısal sistem geliştirme metodolojileriyle birlikte ortaya çıkan klasik çevrim yöntemi; çağlayan (waterfall), şelale modeli, büyük tasarım modeli (grand design) ya da geleneksel model olarak adlandırılır ve **sistematik olarak ilerleyen ardışık bir yaklaşım**la yazılım geliştirilmesini sağlar.
- Diğer bir deyişle, bu modelin temel özelliği belirtilen yaşam döngüsü adımlarını **baştan sona bir kez izleyerek** gerçekleştirir.
- Bu modelde tüm kullanıcı/müşteri gereksinimleri belirlenir, talepler tanımlanır ve buna göre tasarım yapılır; gerçekleştirim sonunda da birimler tümleştirilir

YAZILIM YAŞAM DÖNGÜSÜ

- Daha sonra sistem sınanarak teslim edilir ve bakım aşaması başlatılır.
- Özellikle iyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir model olarak bilinmektedir. Klasik model, belgeleme işlevini ayrı bir işlev olarak ele almaz ve üretimin doğal bir parçası olarak görür. Ancak, bu model son yıllarda çeşitli eleştirilere uğramıştır.

YAZILIM YAŞAM DÖNGÜSÜ



Dr. Öğretim Üyesi Yüksel BAL

YAZILIM YAŞAM DÖNGÜSÜ

Klasik çevrimde;

- Yazılım geliştirme süreçlerinde adım adım ileri veya adım adım geri yönde hareket edilir,
- Şayet test aşamasında bir problem testpit edilirse; gerçekleştirim -> tasarım -> çözümleme şeklinde yeri yönde aşamalara dönülür.
- Bu da büyük zaman kayıplarına yol açar ki, bu metodoloji kullanılırken, personelin bilgi düzeyi, tecrübe ve dikkati çok yüksek olmalıdır,
- Bu metodolojinin kullanılacağı BT firmasının iş yoğunluğu da az olmalıdır.
- Gerçek projeler seyrek olarak ardışık sıra izlerler, çoğunlukla çevrim tekrarı yaşanır bu nedenle de maliyet ve teslim süresi yükselir.

YAZILIM YAŞAM DÖNGÜSÜ

- Müşterinin tüm talepleri bir defada ve açıklıkla tanımlayabilmesi çoğu zaman mümkün değildir. Bu durumda daha projenin başında belirsizlikler ortaya çıkar.
- Müşteri, ürünün tamamı bitinceye kadar beklemek durumundadır. Çalışan bir sürümün ilk ortaya çıkıştı çevrim içinde çok ileride olabilir. Müşterinin hayal kırıklığına uğraması ise projenin başarısızlığı demektir.
- İlk kez çalışılan bir konuda bu çevrim yöntemini izlemek geliştirme süresi kestiriminde yanılgilara neden olur. Daha önce hiç bilinmeyen, araştırılması yapılmamış konularda çalışma gereksinimi ortaya çıkabilir, çözülmesi gereken problemlerin halledilmesi zaman alabilir.

YAZILIM YAŞAM DÖNGÜSÜ

- Geliştirici personel genelde kod yazmaya eğilimlidir. Bu nedenle, çözümleme, tasarım gibi süreçlerden geçmeye alışık olmayan personelin heveslendirilmesi güç olabilir.

Belirtilen sorunlara rağmen, klasik çevrim yazılım mühendisliği alanında diğer teknikler için bir kalıp oluşturmaktadır.

Klasik çevrim süreç modeli özellikle;

- İyi tanımlanmış,
- Talepleri kesinleşmiş ve netleşmiş,
- Fazla zaman alması beklenmeyen, (küçük ölçekli)
- Bilgili ve tecrübeli personelin bulunduğu,

organizasyon ve projeler için uygun bir yöntemdir.

YAZILIM YAŞAM DÖNGÜSÜ

5. Spiral Modeli

Spiral model, daha çok bir üst model ya da tanımlama modeli olarak görülmektedir. Spiral modeli temel olarak alan metodolojilerin ortak özellikleri:

- Yenilemeli ya da artımsal bir yaklaşım içermeleri
- Prototip yaklaşımını ön plana çıkarmaları

Spiral model hem klasik çevrim hem de prototipleme yöntemlerinin iyi yönlerinin birleştirilmesiyle oluşturulmuştur. Bu metodolojide;

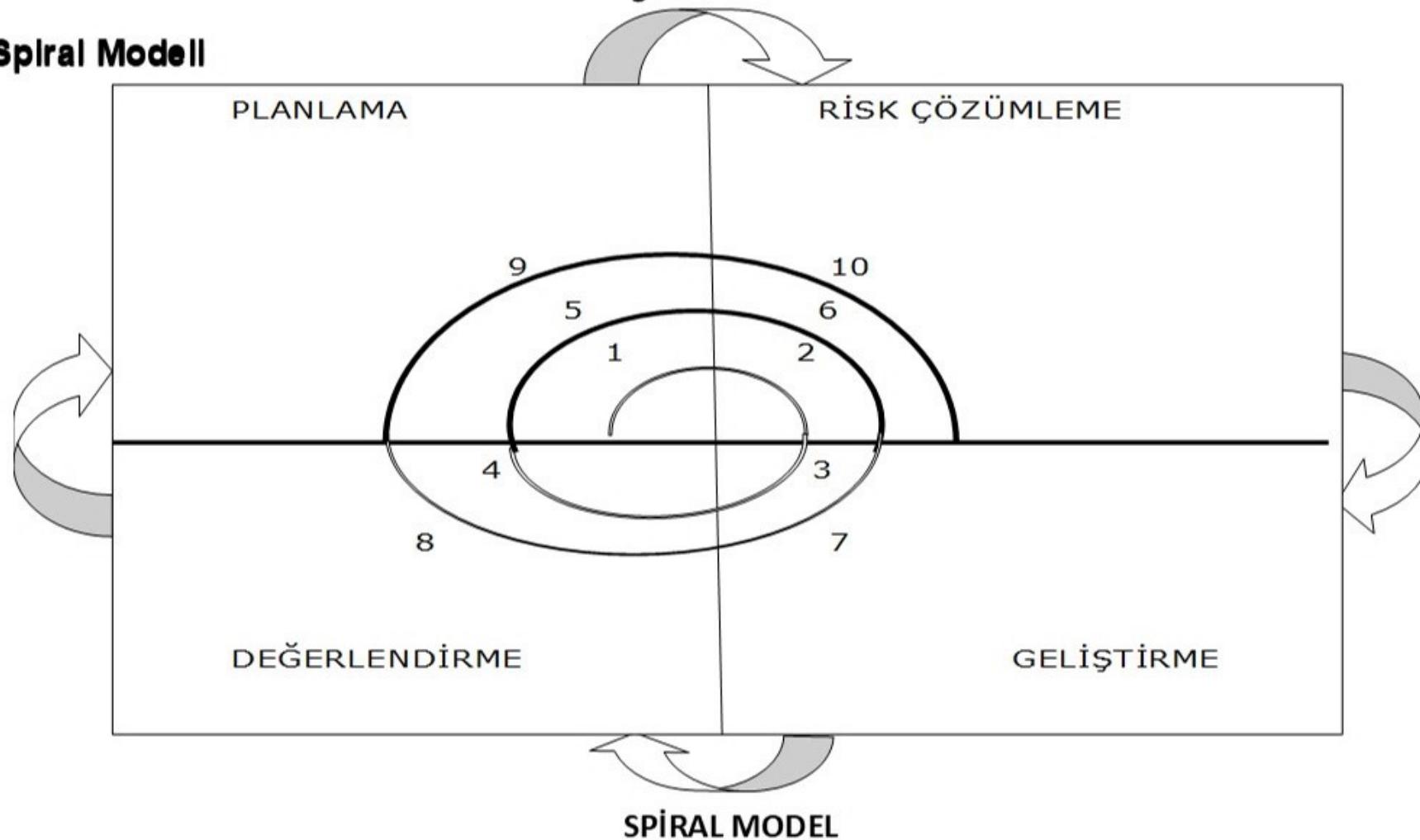
- Risk çözümleme, spiral modelde ön plana çıkarılmıştır.

YAZILIM YAŞAM DÖNGÜSÜ

- Kullanıcının kesin olan gereksinimlerinin bir kısmı belirlenir, bunlardan bir kısım talepler tanımlanır, önce bunların gerçekleştirimi yapılır,
- Ortaya çıkan ürünün testi yapılarak teslim edilir.
- Daha sonra sistemin geri kalanı artımlar ve sürümler halinde geliştirilip teslim edilir.
- Spiral model genel olarak ard arda tekrarlanan dört aşamadan meydana gelmiştir. Bu aşamalar:

YAZILIM YAŞAM DÖNGÜSÜ

5. Spiral Model



Dr. Öğretim Üyesi Yüksel BAL

YAZILIM YAŞAM DÖNGÜSÜ

- Amaçların belirlendiği, olası seçeneklerin ve kısıtlamaların değerlendirildiği *planlama* aşaması,
- Diğer yöntemlerde bulunmayan risklerin tanımlandığı ve olası çözüm yöntemlerinin irdelendiği *risk çözümlemesi* aşaması,
- Ürünün geliştirildiği *üretim* aşaması,
- Geliştirilen ürünün müşteriyle beraber incelendiği, sınandığı *değerlendirme* aşaması.

Bu aşamalar en küçükten başlayıp gittikçe büyüyerek ürünün tamamlanmasına kadar tekrar eden bir çevrim halinde olduğundan spiral modeli adını almıştır. Bu dört aşamada aşağıda belirtilen çalışmalar yapılır:

YAZILIM YAŞAM DÖNGÜSÜ

- Spiralin başladığı ilk çeyrek içinde ilk taleplerin toplanması ve buna göre proje planlaması yapılır.
- İkinci çeyrekte, ilk tanımlanan taleplere göre bir risk çözümlemesi yapılır.
- Üçüncü çeyrekte, risk çözümlemesi sonucunda ortaya çıkan taleplerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleme yöntemi kullanılır.
- Gerekirse benzetim (simülasyon) veya diğer modellemeler kullanılarak taleplerin daha sağlıklı tanımlanmasına çalışılır.
- Dördüncü çeyrekte, müşteri ortaya çıkan ilk prototipi inceleyerek değerlendirme yapar, önerilerde bulunur.

YAZILIM YAŞAM DÖNGÜSÜ

- Bu şekilde tamamlanan ilk döngü, bir sonraki döngü için girdi oluşturur.
- Spiralin üçüncü çeyreğinde bulunan geliştirme aşamasında mutlaka klasik çevrim ya da prototipleme gibi bir yöntem kullanılmalıdır.
- Spiralin merkezinden uzaklaştıkça bu aşamadaki geliştirme işleri daha da artar.
- Spiral modeli, klasik çevrimi geliştirme için kullanmakta, prototipleme yoluyla da riskleri en aza indirmeyi amaçlamaktadır.
- Evimsel bir yaklaşım olarak, müşteri ve geliştiricinin her evrim sırasında beraberce riskleri anaması ve önlemler almasını sağlamaktadır.

YAZILIM YAŞAM DÖNGÜSÜ

Özellikle büyük ve uzun zaman alan projelerde spiral model oldukça başarılı sonuçlar vermektedir. Spiral modelin uyarlamaları:

- VP modeli
- Evrimsel Geliştirme Modeli
- Evrimsel Prototipleme Modeli
- Artımsal Geliştirme Modeli
- Araştırmaya Dayalı Geliştirme Modeli

6. Yeni Teknikler

- Özneye Yönelik Geliştirme
- Bileşen Tabanlı Geliştirme
- Özelliğe Yönelik Programlama
- Uç Programlama

YAZILIM YAŞAM DÖNGÜSÜ

7. Uç Programlama

Uç programlama (Extreme Programming-XP) son zamanlarda ortaya çıkan ve yayılmakta olan bir yazılım geliştirme tekniğidir.

- Yakın bir süre önce bazı büyük şirketlerin uygulamaya koydukları bu yeni yaklaşım, inançlı ve disiplinli bir çalışmayla müşteri memnuniyetinin üzerinde duran, müşterinin istediği yazılımı istediği zaman elde edebilmesini sağlamak üzere tasarlanmış bir yazılım geliştirme yöntemidir.
- Değişen müşteri istekleri ürünün yaşam çevrimi sırasında mutlaka karşılanır.
- Bu yöntem, nitelikli yazılımın oluşturulması için yöneticilerin, müşterinin ve geliştiricinin bir ekip halinde beraberce çalışması esasına dayanır.

YAZILIM YAŞAM DÖNGÜSÜ

Basit fakat etkin bir gurup geliştirme yöntemi kullanılır. XP ile bir yazılım projesi dört ilkeye bağlı olarak geliştirilir:

- *İletişim*: XP programcılar, müşteriyle ve diğer programcılarla sürekli haberleşirler.
- *Basitlik*: XP programcılar, tasarıımı son derece basit ve anlaşılır yaparlar.
- *Geri besleme*: Programcılar ilk günden itibaren yazılımı test ederek geri besleme sağlarlar. Sistemi, müşteriye mümkün olan en kısa zamanda teslim ederler ve talep edilen değişiklikleri hemen yanında uygularlar.
- *Cesaret*: Hızlı geliştirme, teslim ve değişiklik anlayışıyla XP programcılar değişen taleplere ve teknolojilere yanında cevap verme cesaretine sahiptirler.

YAZILIM YAŞAM DÖNGÜSÜ

XP programlama yönteminde, küçük geliştirme işleri kendi başlarına fazla bir şey ifade etmezken, küçük çalışmaların birleşmesi sonucu asıl ürün ortaya çıkar.

Bu çalışma şekli klasik yazılım geliştirme yöntemlerinden ayrılmakta, programlama yapıldıkça gelişme sağlanması ilkesine dayanmaktadır.

XP, geliştirme sırasında talepleri sürekli değişen problem alanları için geliştirilmiştir. Müşteri sistemin ne yapması gerektiği hakkında kesin bir fikre sahip değildir.

Böyle sistemlerin geliştirme süreçlerinde iyi bilinen tek şey taleplerin dinamik olarak değiştiğidir.

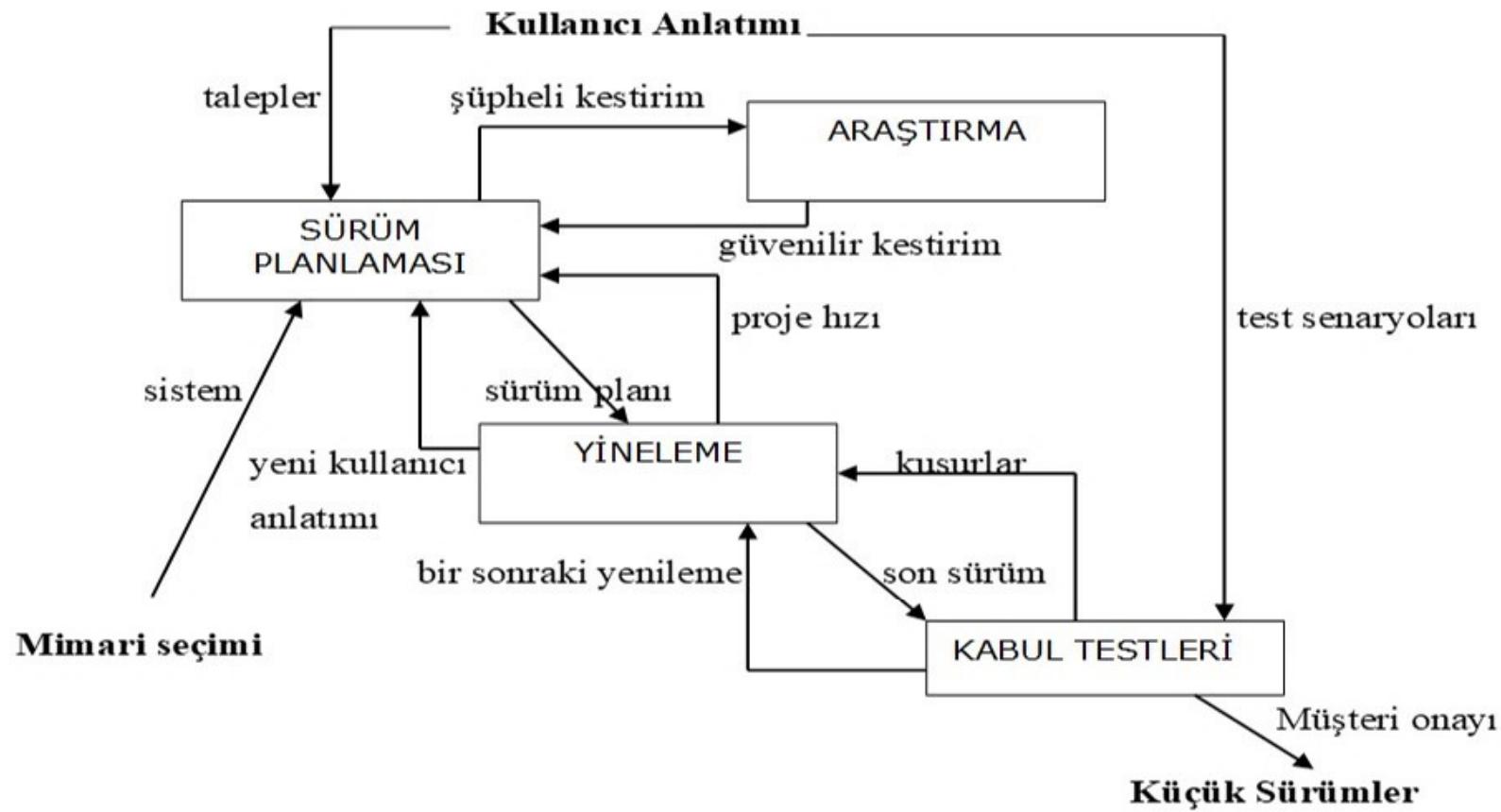
YAZILIM YAŞAM DÖNGÜSÜ

Düzenli bir yazılım yaşam döngüsü, diğer metodolojilerin başarısızlığa uğrayabildiği bu durumda XP başarılı olur. XP, riskin yüksek olduğu durumlarda da yarar sağlar.

Müşteri belirli bir tarihte yeni bir sistem istiyorsa risk oldukça yüksektir. Eğer bu yeni sistem yazılım geliştirme gurubu için yeni ise risk daha da büyütür.

Hatta, bu sistem tüm yazılım endüstrisi için yeni sayılabiliriyorsa risk daha da büyür. XP ile bu risk azaltılıp, başarı şansını arttırılabilir.

YAZILIM YAŞAM DÖNGÜSÜ



Uç programlama

Dr. Öğretim Üyesi Yüksel BAL

YAZILIM YAŞAM DÖNGÜSÜ

- XP, iki ila on kişilik programcı grupları için ortaya çıkmıştır. Büyük projeler için sayının yirmiye hatta otuzaya çıktığı da olabilir. Fakat çok yüksek sayıda personel ile XP kullanılamaz. Programcıların çok tecrübeli olmalarına da gerek yoktur.
- XP için geliştirme ekipleri kurulur. Bu ekiplerde yalnızca **geliştiriciler değil**, aynı zamanda **yöneticiler** ve **müşteri** de yer alır ve yan yana çalışırlar.
- Yazılımın geliştirilmesi sırasında tartışarak talepleri ve kapsamı belirlerler, takvim oluştururlar ve test planaması yaparlar.
- Test edilebilirlik ve üretkenlik son derece önemlidir. Geliştirilen her yazılım biriminin testi mümkün olmalı, diğer yöntemlerle kıyaslanamayacak verimlilik elde edilebilmelidir.

YAZILIM YAŞAM DÖNGÜSÜ

- Ancak, ne olursa olsun tek amaç istenen yazılımı geliştirmektir.
- **XP, düzen gerektirmeyen bir yöntem olarak algılanmamalı, belgelendirme, düzenlenşim ve nitelik güvence etkinlikleri yine yürütülmelidir.**

YAZILIM YAŞAM DÖNGÜSÜ

UÇ PROGRAMLAMA ÖZELLİKLERİ:

Planlama: Her döngüye (tekrar adımına) basit bir planla başlanır ve gerekli oldukça bu planda değişikliğe ve iyileştirmeye gidilir.

Sürümler: Her döngüde çalışan bir sürüm elde etmek temel hedeftir. Her döngü (tekrarlanma süresi) birkaç haftadan daha uzun olmamalıdır.

Basit tasarım: Tasarım mümkün olduğunca basit tutulmalı, gerektiğinde kod düzenleme / düzeltme yapılmalıdır.

Çift programlama: Kodlama aynı anda 2 programcı tarafından yapılmalıdır.

Sürekli bütünlendirme: Her kod parçası tamamlandığında ana programa ilave edilmelidir (büütünleştirilmelidir)

YAZILIM YAŞAM DÖNGÜSÜ

UÇ PROGRAMLAMA ÖZELLİKLERİ:

Belirli çalışma saatı: Geliştirme ekiplerinin verimli çalışabilmesi için aşırı çalışma saatlerinden kaçınılmalıdır.

Müşteri ile yakın çalışma: Müşterinin tam zamanlı olarak geliştirme ekibiyle aynı ekip içerisinde çalışması, yakın temas halinde bulunması veya istendiğinde sürece her an dahil olabilmesi gereklidir.

Kodlama standartları: Tanımlama, kaynak kod düzenleme ve dokümantasyon için kullanılan standartların tüm ekip tarafından benimsenmesi çok önemlidir.

YAZILIM YAŞAM DÖNGÜSÜ

YENİ YAKLAŞIMLAR - AGILE (ÇEVİK) YÖNTEMLER

- Agile; bir Proje Yönetimi yaklaşımı olup bir metodoloji değildir. Şemsiyesi altındaki yönetimsel ve teknik araçlar ile nasıl proje yönetimi yapılabileceğine dair bir yaklaşımdır.
- Ekiplerin büyüklüğüne, mevcut çalışma şekline ve ortaya çıkacak ürünün dinamiklerine göre farklı araçlar kullanılabilir.
- Bu araçlar;
 - ✓ Dynamic System Development Methodology (DSDM),
 - ✓ Scrum
 - ✓ Kanban
 - ✓ Extreme Programming (XP),
 - ✓ Test Driven Development (TDD) ,
 - ✓ Feature Driven Development (FDD),
 - ✓ Lean Development

gibi listelenebilir.

YAZILIM YAŞAM DÖNGÜSÜ

YENİ YAKLAŞIMLAR - AGILE (ÇEVİK) YÖNTEMLER

Sonuç Olarak Çevik Yöntemler;

- Bu araçlar uygulamada değer yaratacağını düşündüğümüz minimum kapsamı hedefleyerek **pazara hızlı çıkmamızı** sağlar.
- Sürekli teslim süreci ile değişen pazar koşullarına uyum sağlamamıza, düzenli değerlendirmelere müşteriyi dahil ederek **doğru ürünü üretmemize** yardımcı olur.
- Bunların yanında, süreçte yarattığı şeffaflık ile iyileşmeye açık noktaları tespit etmemizi ve iyileştirmemizi, sonucunda da **maksimum üretkenlikte değer elde etmemizi** sağlar.

YAZILIM YAŞAM DÖNGÜSÜ

YENİ YAKLAŞIMLAR - AGILE (ÇEVİK) YÖNTEMLER

“Agile olmak” aslında bir düşünüş biçimidir.

- Verimli ve üretken bir şekilde değişime adapte olup müşteriye yüksek değer üretmeye çalışmaktadır.
- Bu düşüncenin temeline konulan bazı değerler var;
Bunlar ekip olmak, iletişim, değişime adapte olabilmek hatta değişimi yaratan taraf olabilmektir.
- Müşteriyi ayrı bir taraf olarak değil, hedefe birlikte koşan ekibin bir parçası olarak görmektir.
- Yönetimsel ve teknik araçları kullanmak Agile yaklaşımı sağlasa da asıl başarı zihniyet değişimi ile sağlanabilir.

YAZILIM YAŞAM DÖNGÜSÜ

Tekniklerin Birleştirilmesi

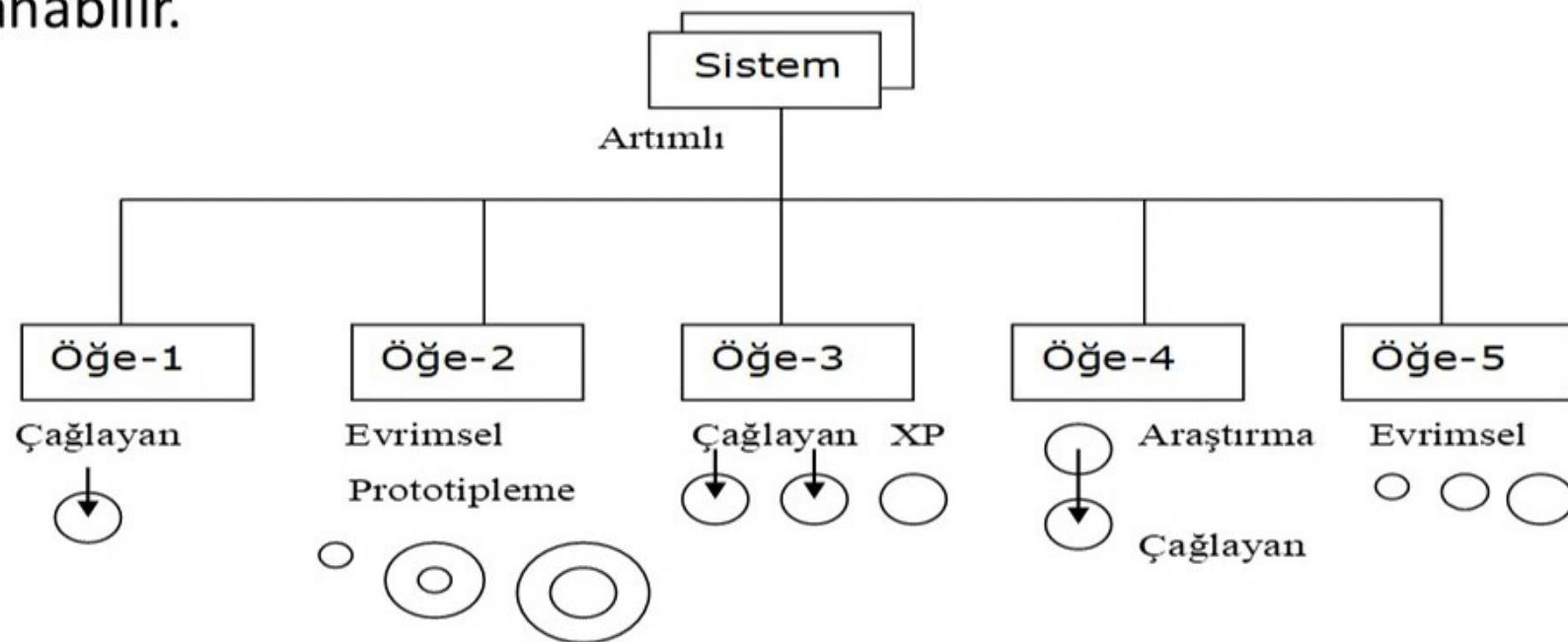
Çeşitli durumlarda ve özellikle çok büyük projelerde daha önceki slaytlarda belirtilen yöntemlerin birçoğu bir arada da kullanılabilmektedir. Böylece herbirinin sahip olduğu güçlü yönler birleştirilerek daha etkin bir yazılım geliştirme süreci oluşturulabilir.

Önceki slaytlarda kısaca bahsedilen metodolojiler farklı şekillerde birlikte kullanılarak büyük bir projenin geliştirilmesi gerçekleştirilebilir.

YAZILIM YAŞAM DÖNGÜSÜ

Tekniklerin Birleştirilmesi

Aşağıdaki şekilde birleştirilmiş teknik kullanılarak projenin herbir parçası için ayrı bir metodoloji kullanılmak suretiyle büyük bir sistemin geliştirilmesi sağlanabilir.



Metodoloji'nin Önemi ve Seçimi

BT firmalarında, yazılım geliştirme süreçleri için metodoloji seçimi çok önemlidir.

Genellikle uygulamalarda, yazılımcı(lar) müşteri ya da kullanıcılar ile görüşerek doğrudan geliştirmeye başlarlar.

Oysaki, hem yazılım geliştirmede kullanılacak doğru metodoloji seçilmesi ve buna bağlı olarak yazılım geliştirme süreçlerinin atlanmadan standartlara uygun şekilde gerçekleştirilmesi bilgi teknolojilerinde en önemli konulardan biridir.

Metodoloji seçilirken bazı ayrıntılara da dikkat edilmesi ayrıca önem taşımaktadır.

Metodoloji'nin Önemi ve Seçimi

Metodoloji seçimi; kalite yönetimi, proje yönetimi ve ilgili ekiplerle birlikte yapılmalıdır. Seçim yapılırken dikkat edilmesi gereken önemli kriterlerden bazıları aşağıdaki gibi özetlenebilir:

- **Bilgi Teknolojileri Firmasının yazılım geliştirme talep yoğunluğu,**
- **Proje Süreleri,**
- **Personel yetkinlik ve bilgi düzeyi (Analist, Tasarımcı, Developer, tester vb),**
- **Teknik altyapı yeterliliği (donanım ve yazılım araçları)**
- **Müşteri profili ve talepleri,**
- **Çalışılan sektör,**
- **Süreçler, iş akışları ve organizasyon yapısı**

Bazı kriterler dikkate alınmadan metodoloji seçimi yapmak, **gelistirme zamanını uzatır, maliyetleri arttırmır ve müşteri memnuniyetsizliklerine neden olur.**

YAZILIM YAŞAM DÖNGÜSÜ

Metodoloji bir BT projesi ya da yazılımın yaşam döngüsü aşamaları boyunca kullanılacak ve birbiriyle uyumlu yöntemler bütününe içerir.

Herhangi bir metodoloji;

- Bir süreç modeli
- Belirli sayıda belirtim yönteminden oluşur.

Günümüzde uygulamada 100'den fazla metodoloji kullanılmaktadır. Birçok firma belirli bir süreç modelini temel alarak kendi metodolojilerini geliştirmekte ve uygulamaktadır. Metodolojiler genellikle çağlayan (waterfall) veya helezonik (spiral) modeli **temel** almaktadır.

YAZILIM YAŞAM DÖNGÜSÜ

Bir metodolojide bulunması gereken temel özellikler:

- Detaylandırılmış bir süreç modeli
- Detaylı süreç tanımları
- İyi tanımlanmış üretim yöntemleri
- Süreçler arası ara yüz tanımları
- Ayrıntılı girdi tanımları
- Ayrıntılı çıktı tanımları
- Proje Yönetim Modeli
- Konfigürasyon Yönetim Modeli
- Maliyet Yönetim Modeli
- Kalite yönetim modeli
- Risk Yönetim Modeli
- Değişiklik yönetim modeli
- Kullanıcı arayüz ve ilişki modeli
- Standartlar

Yukarıdaki maddelerle ilgili bağımsız kuruluşlar (IEEE, ISO, SEI, ESI vs) ve kişiler tarafından sürekli yeni standard ve kılavuzlar üretilmektedir. Bu yüzden kullanılan süreç modelleri ve belirtim yöntemleri zaman içerisinde değişmekte ve gelişmektedir.

YAZILIM YAŞAM DÖNGÜSÜ

Yazılım yaşam döngüsü çevrim süresini azaltmak için:

- **Müşteri gereksinimleri önceden ve somut bir şekilde tesbit edilmelidir,**
- **Yazılımın tekrar kullanımı arttırılır,**
- **Değişiklik talepleri azaltılır,**
- **Süreçler iyileştirilir ve basitleştirilir,**
- **Çalışma ortamında iyileştirmeler sağlanır,**
- **İş için en bilgili ve tecrübeli personel kullanılır,**
- **Sorunlara aktif bir şekilde yaklaşılır,**
- **Standartlar kullanılır,**
- **Kod ve algoritma karmaşıklığı engellenir.**

Kaynaklar

- David Gustafson, 'Software Engineering'
- M. Erhan Sarıdoğan, 'Yazılım Mühendisliği',
- Ali Arifoğlu, Ali Doğru, 'Yazılım Mühendisliği'
- Oya Kalıpsız, Ayşe Buharalı, Ayşe Biricik, 'Sistem Analizi ve Tasarımı'
- Yüksel Bal, 'Yazılım Projesi Geliştirme' Ders Notları,
- Yüksel Bal, 'Yazılım Mühendisliği ve Sistem Analizi' Ders Notları,
- Çeşitli İnternet Kaynakları,