MOBİL UYGULAMALARDA GÜVENLİK RİSKLERİ VE ATAKLAR

Olası saldırıları önlemek ve mobil cihazlarda güvenliği sağlamak için aşağıdaki adımlar izlenmelidir.

- Hazırlık,
- Bilgi Toplama,
- İş Parçacığı Modelleme,
- Zafiyet Analizi,
- Saldırıları/Zararlı Yazılımı belirleme,
- Güçlü Şifreleme Yöntemleri Kullanma,
- Saldırıları önleme,
- Sistem açıklarını düzenli olarak kontrol etme.

Bu adımlar izlenerek mobil cihazlar için test geliştirilebilir.

Saldırı Çeşitleri

• **İzinsiz Dinlemek (Sniffing):** Her türlü veri aktarımının izinsiz üçüncül bir kişi/sistem tarafından dinlenmesi ve/veya elde edilmesidir. Amacı; şifreleri, E-posta metnini, transfer edilen dosyaları yakalamaktır.
• **Toplu Gönderim (Spamming):** Kullanıcının isteği dışında, kullanıcıya ait olan bir cihaza veya adrese gönderilen sahte iletiler ile bilgilere ulaşmayı hedefler. Örneğin istenmeyen E-posta, istenmeyen MMS mesajı gibi
• **Yanıltma (Spoofing):** DNS zehirlenmesi olarak da adlandırılır. DNS sunucusunun ön bellek bilgilerine yeni bir veri eklenerek veya aradaki verileri değiştirerek sunucunun isminin yanlış IP adresleri göndermesine ve trafiğin başka bir bilgisayara yönlendirilmesine neden olan bir saldırıdır. Genel olarak saldırganın ağ üzerindeki bir sağlayıcıya erişip onun üzerinden bir başka kurbana saldırmasıdır.
• **Oltalama (Phishing):** Bankalar veya finans şirketleri tarafından gönderilmiş gibi görünen ve acil konular, önemli bilgiler içeriyormuş gibi duran sahte postalardır. E-posta ve web site oltalama işleminin birleşimidir. • **Yönlendirme (Pharming):** Erişilmeye çalışılandan farklı bir illegal web sayfasına yönlendirmekle gerçekleştirilen bilişim suçudur.
• **Veri Sızıntısı (Data Leakage):** Uygulamalardan kısmi veri alınarak yapılan saldırıdır.
• **Hizmet Engelleme (DoS) Atakları:** Saldırganın daha önceden tasarlanmış olduğu makine üzerinden hedefe saldırı yaparak, karşı tarafın sisteminin kimseye hizmet veremez hale gelmesini sağlamayı amaçlayan bir saldırı çeşididir. Örneğin, bataryaya aşırı yüklenme (exhausting), radyo frekansına karıştırma (jamming) gibi.

İş Parçacığı (Thread) Modeli

İş parçacığı üç katmana ayrılır.

- Uygulama Katmanı.
- Haberleşme Katmanı.
- Kaynak Katmanı.

İş Parçacığı (Thread) Modeli

- **Uygulama Katmanı;** Akıllı telefon ve tabletlerdeki tüm uygulamaları kapsar. Kötücül yazılım normal bir uygulama gibi davranır ve kullanıcının uygulama olarak indirmesini sağlar.
- **Haberleşme Katmanı;** Kötücül yazılım mobil cihaza bağlantı kanalları vasıtasıyla girer.
- **Kaynak Katmanı;** Mobil cihazın en önemli katmanıdır. Önemli bilgiler burada saklanır. Kötücül yazılım buradaki bilgileri kontrol ederek cihazı manipüle eder.

Zafiyet Analizi

Adli Bilişim: Mobil cihazdaki veriyi inceleyerek değerlendirmeleri için kullanıcıya birçok imkan sunar. Kullandığı 2 araç vardır; “Micro Systemation’s XRY” ve “Cellecrite UFED Touch Ultimate”. Bu araçlar veri silinmiş olsa bile veriden mantıksal ve fiziksel çıkarımlar yaparlar. Mobil cihaz tam taramaya/incelemeye açıkken, veri ile ilgili sınırsız çıkarım yapılabilir.

Statik Analiz : Uygulamayı çalıştırmadan uygulamanın analiz edilmesidir. Statik analiz uygulamanın kodunda bulunan zararlı yazılımları bulmak için kodu inceler. Uygulamak için kullanılabilecek araçlar, “Android Reverse Tools”, “Static Android Analysis Framework”. Ayrıca statik analiz geri derleme yaparak kodu analiz eder.

Dinamik Analiz : Uygulamayı çalıştırarak uygulamanın analiz edilmesidir. Ağ trafiğini göstererek kötü niyetli aktiviteleri ortaya çıkarmayı amaçlar. Dinamik Analiz programın işleyişine dayalı olduğundan geri derleme yapmaya gerek yoktur. Kötü niyetli yazılımlar gizlenemediği için statik analize göre daha iyi çalışır.

Fonksiyonel Testler

Geliştirilen uygulamanın kullanıcı ihtiyaçlarını ne derecede karşıladığı ölçülür. Arayüz ve uygulamanın genel durumundan yola çıkılarak kullanıcılar için kullanışlı olup olmadığını belirlemeye yardımcı testlerdir.

Fonksiyonel Testler

Birim Testi : Özel fonksiyonlar veya kod bileşenleri test edilir. Bu testin yapılabilmesi için program kodunun mimarisinin ayrıntılı bir şekilde bilinmesi gerekir.

Regresyon Testi : istemde gerekli ve son değişiklikler yapıldıktan sonra gerçekleştirilen testlerdir. Bu sayede, daha önceki testlerde ortaya çıkan sorunların giderildiğinden ve yeni hatalar yapılmadığından emin olunur.

Fonksiyonel Testler

Sızma Testi : Sızma Testi “Penetrasyon Test” olarak da bilinir. Firmaların bilgi işlem sistemlerini oluşturan altyapıya ve uygulamalara bir saldırganın kullanacağı yöntemler kullanılarak saldırılır ardından müdahaleler sonucunda güvenlik açıkları tespit edilir, bu açıklar ile sistemlere sızılmaya çalışılır, bu açıkların nelere sebep olabileceği incelenir. İşlem bittiğinde sonuçları raporlanır. Mobil sızma testleri cihazın içeriğini düzgünce test etmek için çeşitli ayarlar bulundurur, test sonuçlarını inceler

Fonksiyonel Testler

Laboratuvar Testi: Geliştirilen uygulamaların farklı operatörlerde ve farklı ağ bağlantılarında düzgün çalışıp çalışmadığı kontrol edilir. Genellikle ağ üzerinden yürütülürler ve ağın benzetimi yürütülür. Mobil uygulamaların; her koşulunda aynı tutarlılıkta ve hızda çalışması önemlidir

Fonksiyonel Testler

Performans Testi : Belirli koşullar altında uygulamanın cihazın performansına etkisini ve nasıl çalıştığını kontrol etmek için yapılan testtir. Bunlar düşük çekim gücü, düşük batarya, kısıtlı hafıza gibi. Hem kullanıcı, hem de geliştirici tarafından incelenerek değerlendirme yapılır

Fonksiyonel Testler

Kesme Testi : Kullanılan mobil cihazın temel işlevlerinin baz alındığı ve uygulamanın çalışmaya ara verdiği süreç incelenir. Uygulama yeniden çalıştırıldığında uygulamanın işlevini yerine getirip getirmediği test edilir.

Fonksiyonel Testler

Kullanılabilirlik Testi: Geliştirilen uygulamanın amacına uygun olup olmadığını, kullanıcılar tarafından ne kadar tercih edildiğini belirlemek için yapılır. Aynı zamanda, uygulamanın ticari başarısının ölçülmesinde de önemli rol oynar. Sezgisel (Heuristic), değerlendirici (evaluation), bilişsel gözden geçirme (cognitive walkthrough) ve sesli-düşünme (thinking-aloud study) araştırmacıların kullandıkları bazı yöntemlerdir

Fonksiyonel Testler

Laboratuvar Testi : Geliştirilen uygulamaların farklı operatörlerde ve farklı ağ bağlantılarında düzgün çalışıp çalışmadığı kontrol edilir. Genellikle ağ üzerinden yürütülürler ve ağın benzetimi yürütülür. Mobil uygulamaların; her koşulunda aynı tutarlılıkta ve hızda çalışması önemlidir

Kişisel ve kurumsal verileri korumak için alınması gereken önlemler.

Kişisel verilerin korunmasına yönelik önlemler

1.Veri şifreleme: Kişisel veriler, depolanırken ve iletilirken şifrelenmelidir. Bu, verilerin yalnızca yetkilendirilmiş kişiler tarafından erişilebilmesini sağlar.

- **Endüstri standartlarına uygun şifreleme yöntemleri:** AES (Advanced Encryption Standard) gibi güçlü şifreleme algoritmalarının kullanılması önerilir.

2. Çift Faktörlü Kimlik Doğrulama:

- **Ek güvenlik katmanı:** Hesaplara giriş yaparken yalnızca şifre değil, ek bir doğrulama adımı (SMS, e-posta doğrulaması, uygulama tabanlı doğrulama) gereklidir.
- **Hesap güvenliği:** Kişisel hesapların, özellikle e-posta, bankacılık ve sosyal medya hesaplarının güvenliği sağlanmış olur.

Kişisel verilerin korunmasına yönelik önlemler

3.Güçlü Parolalar ve Parola Yönetimi:

- **Karmaşık parolalar:** Parolalar, en az 8-12 karakter uzunluğunda, büyük harf, küçük harf, rakam ve özel karakterler içermelidir.
- **Parola yöneticileri kullanmak:** Parolaların güvenli şekilde saklanması ve yönetilmesi için parola yöneticileri kullanılabilir.
- **Düzenli parola değişimi:** Parolalar, belirli aralıklarla değiştirilmelidir.

4. Erişim Kontrolü ve Yetkilendirme:

- **Azami yetki prensibi (Least Privilege):** Kişisel verilere erişim yalnızca gerektiği kadar verilmelidir. Gereksiz erişimler engellenmelidir.
- **Erişim seviyeleri:** Kullanıcılar için uygun erişim seviyeleri belirlenmeli ve kişisel verilere sadece yetkilendirilmiş kişiler erişebilmelidir.

Kişisel verilerin korunmasına yönelik önlemler

5. Veri Maskelenmesi ve Anonimleştirilmesi ;

- **Veri maskelenmesi:** Kişisel veriler sadece belirli durumlarda erişilebilecek şekilde maskelenmeli, geriye doğru gizli bilgiler saklanmalıdır.
- **Anonimleştirme:** Özellikle anketler, araştırmalar ve veri tabanlarında, kimlik belirleyici bilgilerin kaldırılması gerekir.

Kişisel verilerin korunmasına yönelik önlemler

6. Gizlilik Politikaları ve Kullanıcı Onayı;

- **Gizlilik politikaları:** Kişisel verilerin nasıl toplandığı, kullanıldığı ve saklandığına dair açık ve şeffaf bir gizlilik politikası oluşturulmalıdır.
- **Kullanıcı onayı:** Kişisel verilerin toplanmasından önce kullanıcıdan açık rıza alınmalıdır. Rıza belgesi oluşturulmalıdır.

Kişisel verilerin korunmasına yönelik önlemler

7. Veri İhlali Durumunda Hızlı Müdahale;

- **Veri ihlali bildirim prosedürleri:** Eğer kişisel verilere izinsiz erişim sağlanmışsa, kullanıcılar en kısa sürede bilgilendirilmeli ve ihlalin etkileri minimize edilmelidir.
- **Acil müdahale planı:** Olası veri ihlali durumları için bir acil müdahale planı hazırlanmalıdır.

Kişisel verilerin korunmasına yönelik önlemler

8. Fiziksel Güvenlik ;

- **Veri depolama cihazlarının güvenliği:** Kişisel verilerin saklandığı cihazlar fiziksel olarak korunmalı, kaybolma veya çalınma riski en aza indirilmelidir.
- **Cihazların şifrelenmesi:** Özellikle taşınabilir cihazlar (laptop, mobil telefon vb.) şifrelenmeli ve güvenli bir şekilde saklanmalıdır.

Kişisel verilerin korunmasına yönelik önlemler

9. Dijital İmza ve Elektronik Kimlik Doğrulama ;

- **Dijital imzalar:** Elektronik ortamda yapılan işlemlerin doğruluğunu ve güvenliğini sağlamak için dijital imzalar kullanılabilir.
- **Kimlik doğrulama yöntemleri:** Özellikle online platformlarda elektronik kimlik doğrulama işlemleri yapılmalıdır.

Kişisel verilerin korunmasına yönelik önlemler

10. Yazılım ve Sistem Güncellemeleri;

- **Yazılım güncellemeleri:** Kişisel verilerin saklandığı ve işlendiği yazılımlar düzenli olarak güncellenmeli, güvenlik açıkları giderilmelidir.
- **Otomatik güncellemeler:** Güvenlik açıklarını kapatmak için otomatik güncelleme özellikleri etkinleştirilmelidir.

Kişisel verilerin korunmasına yönelik önlemler

11. Veri Silme ve İmha Prosedürleri;

- **Veri silme:** Kullanılmayan ya da gereksiz hale gelen kişisel veriler güvenli bir şekilde silinmelidir. Yalnızca belirli bir süre saklanan veriler, gerekli süre sonunda imha edilmelidir.
- **Fiziksel cihazların imhası:** Eski bilgisayarlar ve cihazlar tamamen silinmeli ya da fiziksel olarak imha edilmelidir.

Kişisel verilerin korunmasına yönelik önlemler

12. Eğitim ve Farkındalık ;

- **Kullanıcı eğitimi:** Çalışanlar ve bireyler, veri güvenliği ve gizliliği konusunda eğitim almalı, kimlik avı (phishing) gibi saldırılara karşı bilinçlendirilmelidir.
- **Farkındalık programları:** Çalışanlar, kişisel verilerin korunması ve veri güvenliği politikaları hakkında sürekli olarak bilgilendirilmelidir.

Kişisel verilerin korunmasına yönelik önlemler

13. Veri Paylaşımı ve Dış Kaynak Kullanımı;

- **Veri paylaşımı:** Kişisel veriler yalnızca gerekli durumlarda ve yetkilendirilmiş kişilerle paylaşılmalıdır.
- **Üçüncü taraf güvenliği:** Dış kaynaklardan hizmet alırken, bu hizmet sağlayıcıların da veri güvenliği standartlarına uyum sağladığından emin olunmalıdır.

Kişisel verilerin korunmasına yönelik önlemler

14. Yasal Düzenlemelere Uyum;

- **GDPR ve KVKK:** Kişisel verilerin korunması için geçerli yasal düzenlemelere (örneğin, Avrupa Birliği'nin Genel Veri Koruma Yönetmeliği – GDPR veya Türkiye'deki Kişisel Verilerin Korunması Kanunu – KVKK) uyulmalıdır.
- **Veri işlemeyle ilgili yasal gereklilikler:** Kişisel verilerle ilgili yasal yükümlülükler yerine getirilmelidir (örneğin, kullanıcı rızası alınması, veri saklama sürelerine uyulması).

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

1. Veri Şifreleme

- **Veri şifrelemesi:** Kurumsal veriler hem depolama alanlarında hem de ağ üzerinden iletilirken şifrelenir. Bu, verilerin kötü niyetli kişiler tarafından erişilememesini sağlar.
- **Endüstri standartlarına uygun şifreleme:** AES (Advanced Encryption Standard) gibi güçlü şifreleme algoritmaları kullanılır.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

2. Erişim Kontrolü ve Yetkilendirme

- **Azami yetki prensibi (Least Privilege):** Çalışanlara yalnızca görevlerini yerine getirmeleri için gerekli minimum erişim verilmelidir. Gereksiz erişimler engellenir.
- **Roller ve izinler:** Erişim seviyeleri belirlenir ve kişisel verilere yalnızca yetkili kişiler erişebilir.
- **Zaman sınırlı erişim:** Belirli süreler için geçici erişim hakları verilerek, süre bitiminde bu erişimler iptal edilir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

3. Çift Faktörlü Kimlik Doğrulama

- **Çift faktörlü doğrulama:** Şirket sistemlerine ve verilere erişim sağlamak için ikinci bir doğrulama adımı kullanılır. Bu, şifreyi takiben SMS, uygulama veya donanım token'ı ile ek bir güvenlik katmanı ekler.
- **Hesap güvenliği:** Çalışanlar, önemli sistemlere giriş yaparken bu güvenlik katmanlarını kullanmak zorundadır.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

4. Düzenli Yazılım Güncellemeleri

- **Güvenlik açıklarının kapatılması:** Sistemlerdeki yazılım güncellemeleri düzenli olarak yapılır. Özellikle güvenlik yamaları, herhangi bir siber saldırıya karşı savunma sağlar.
- **Otomatik güncellemeler:** Güvenlik açıkları hızlıca kapatılmak için yazılımlar otomatik olarak güncellenir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

5. Antivirüs ve Antimalware Yazılımları

- **Zararlı yazılım koruması:** Kurumsal ağlar, cihazlar ve sunucular antivirüs ve antimalware yazılımları ile korunur.
- **Düzenli taramalar:** Cihazlar, ağlar ve sistemler zararlı yazılımlar için düzenli olarak taranır

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

6. Ağ Güvenliği

- **Güvenlik duvarları:** Kurumsal ağlar, dış tehditlerden korunmak için güvenlik duvarları (firewall) ile korunur.
- **VPN kullanımı:** Uzaktan erişim sağlanırken, şifreli bir bağlantı için VPN (Virtual Private Network) kullanılır.
- **Ağ izleme:** Ağa bağlı cihazlar sürekli izlenir ve şüpheli aktiviteler hızlıca tespit edilir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

7. Veri Yedekleme ve Kurtarma

- **Düzenli veri yedeklemeleri:** Veriler düzenli olarak yedeklenir. Bu, veri kaybı durumunda işletmenin hızlıca eski haline dönmesini sağlar.
- **Acil durum kurtarma planı:** Veri kaybı durumunda uygulanacak adımlar ve kurtarma prosedürleri oluşturulur.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

8. Fiziksel Güvenlik

- **Donanım güvenliği:** Sunucular, bilgisayarlar ve diğer cihazlar fiziksel olarak güvenli alanlarda tutulur.
- **Erişim kontrolü:** Veri merkezlerine ve ofislere yalnızca yetkilendirilmiş kişiler girebilir. Cihazlar güvenli bir şekilde saklanır.
- **Cihazların şifrelenmesi:** Taşınabilir cihazlar (laptop, mobil cihazlar vb.) şifrelenerek kaybolma veya çalınma durumunda veri sızması engellenir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

9. Veri Maskelenmesi ve Anonimleştirilmesi

- **Veri maskelenmesi:** Veriler, yalnızca gerekli kişiler tarafından görünür olmalı ve gerektiğinde maskelenmelidir.
- **Anonimleştirme:** Özellikle kişisel veriler anonimleştirilerek, veri ihlali durumunda bile gizlilik korunur.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

10. Çalışan Eğitimi ve Farkındalık

- **Güvenlik eğitimi:** Çalışanlar, kimlik avı (phishing), şüpheli e-postalar ve sosyal mühendislik gibi güvenlik tehditleri konusunda eğitilir.
- **Farkındalık programları:** Veri güvenliği konusunda sürekli eğitimler ve farkındalık programları düzenlenir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

11. Dış Kaynak Kullanımı ve Üçüncü Taraf Denetimi

- **Veri paylaşımı:** Dış kaynaklarla veri paylaşımı, güvenli yollarla yapılmalıdır. Üçüncü taraflarla yapılan sözleşmelerde veri güvenliği şartları net bir şekilde belirtilmelidir.
- **Denetimler:** Üçüncü taraf hizmet sağlayıcıların güvenlik önlemleri düzenli olarak denetlenir ve uyum standartları gözden geçirilir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

12. Veri İhlali İzleme ve Yanıt Verme

- **İzleme sistemleri:** Kurumsal ağ ve sistemler, şüpheli aktiviteler için izlenir. Anomaliler tespit edilir ve anında müdahale edilir.
- **Veri ihlali planı:** Olası bir veri ihlali durumunda, hızlı müdahale için bir plan hazırlanır ve uygulanır.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

13. Gizlilik Politikaları ve Yasal Uyumluluk

- **Gizlilik politikaları:** Kişisel verilerin nasıl toplandığı, işlendiği ve saklandığı hakkında açık ve şeffaf politikalar oluşturulur.
- **Yasal uyumluluk:** GDPR, KVKK ve diğer yerel yasal düzenlemelere uyum sağlanır. Kurumsal verilerin korunması için yasal yükümlülüklere riayet edilir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

14. Veri Silme ve İmha Prosedürleri

- **Veri silme:** Kullanılmayan veya gereksiz veriler güvenli bir şekilde silinir.
- **Fiziksel cihaz imhası:** Eski cihazlar, verilerin kurtarılamayacak şekilde silinmesi veya fiziksel olarak imha edilmesi ile güvenli şekilde yok edilir.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

15. Sosyal Mühendislik ve Kimlik Avı Saldırılarına Karşı Korunma

- **Kimlik avı (phishing) saldırıları:** Çalışanlar kimlik avı saldırıları konusunda eğitilir ve bu tür e-postaların açılmaması gerektiği anlatılır.
- **E-posta doğrulama:** Çalışanlar, e-posta adreslerini doğrulamadan güvenli bilgi paylaşımı yapmamalıdır.

Kurumsal verilerin korunması, şirketlerin ve organizasyonlarının korunmasına yönelik önlemler

16. Dijital İmza ve Elektronik Kimlik Doğrulama

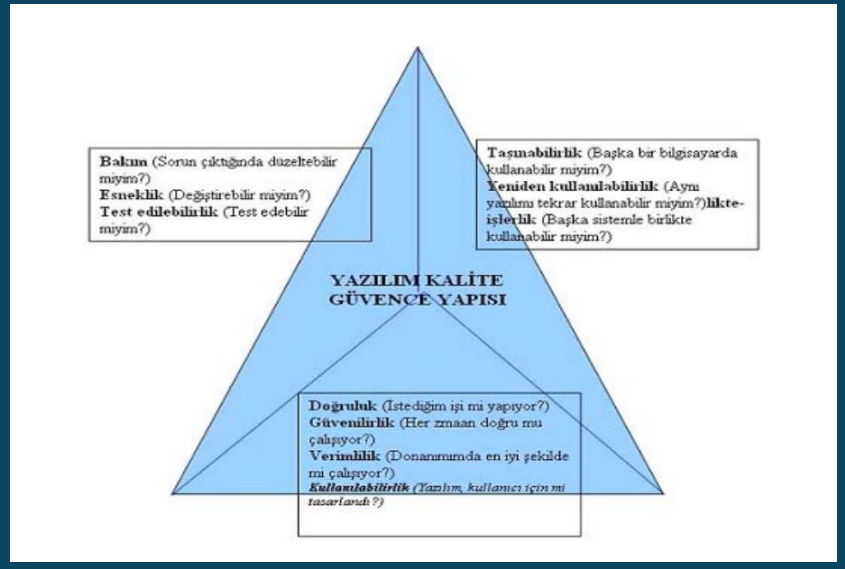
- **Dijital imzalar:** Elektronik belgeler ve işlemler, dijital imzalarla korunarak, kimlik doğrulama sağlanır.
- **Elektronik kimlik doğrulama:** Çalışanların online işlemlerinde kimlik doğrulama yöntemleri kullanılmalıdır.

TÜRKİYE'DEKİ ŞİRKETLERDE EN SIK RASTLANAN GÜVENLİK AÇIKLARI

- Hatalı Kablosuz Ağ Yapılandırması
- Hatalı Yapılandırılmış Sanal Özel Ağ (VPN)
- Sunucuları Web Uygulamalarında SQL Sorgularının Değiştirilebilmesi
- Web Uygulamalarında Başka Siteden Kod Çalıştırma
- Kolay Tahmin Edilebilir Şifrelere Sahip Kullanıcı Hesapları SNMP Servisi Kullanımı
- Güncellemeleri Yapılmamış Web Sunucusu
- İşletim Sistemi ve Uygulamaların Standart Şekilde Kurulması Hatalı Yapılandırılmış Saldırı Tespit Sistemleri
- Güvenlik Duvarı Tarafından Korunmayan Sistemler

YAZILIM KALİTE VE GÜVENLİK AKTİVİTELERİ

YKG aktivitesi yazılım sürecinin başından sonuna kadar uygulanması gereken bir şemsiye aktivitesidir. Yazılım kalitesi kod ortaya çıktıktan sonra üzerinde düşünülmesi gereken bir olgu değildir. YKG aktiviteleri kaliteye bağlı yönetim yaklaşımını desteklemek, etkin yazılım mühendisliği metodlarını öğretmek, formal teknik görüşmelerler düzenlemek, test stratejilerini oluşturmak, yazılım dokümanları kontrol etmek, yazılım geliştirme standartlarına ne kadar uyulduğunu belirlemek ölçmek ve raporlamak amacıyla oluşturulmuştur.



KALİTE GÜVENCE HEDEFLERİ

Yazılım geliştirme, diğer tüm karmaşık geliştirme aktiviteleri gibi risklerle doludur. Bu riskler, teknik veya program ile ilgili olabilir. Bu da, yazılımın beklenildiği gibi olmasını engelleyebilir.
KG'nin hedefi, bu riskleri azaltmak, CIA üçgeninin tamamlanmasını sağlamaktır. Örneğin, kodlama standartları yazılan kod kalitesini belirli bir seviyede tutmak için gereklidir. Eğer herhangi bir standart yoksa oluşan kodun, yeniden kullanılabilirlik ile ilgili gereksinimlere uygun olmaması riski vardır ve bu da kodun üstünde tekrar çalışma yapmayı gerektirir.

YAZILIM GELİŞTİRME MODELLERİ

YETENEK OLGUNLUK MODELİ (CMM - CAPABİLİTY MATURİTY MODEL)

Amerikan Savunma Bakanlığı'nın isteği üzerine Carnegie Mellon Üniversitesi'ne bağlı Yazılım Mühendisliği Enstitüsü tarafından 1986 yılında geliştirilmeye başlanmıştır. Yetenek Olgunluk Modeli (Capability Maturity Model) belirli mühendislik disiplinleri için referans olgunluk modeli sağlar. Bir organizasyon kendi pratik uygulamalarını muhtemel iyileştirmeleri sağlamak için model ile karşılaştırır. CMM özel süreçler (yazılım mühendisliği, sistem mühendisliği, güvenlik mühendisliği) için hedef aşamalar tanımlar. Fakat bu işlemler için rehber dokümanlar sağlamaz. Bu süreçler ne istendiğini tanımlar fakat nasıl olacağını tanımlamaz. “CMM tabanlı değerlendirmeler ürün değerlendirmesinin ya da sistem sertifikasyonunun yerini 18 alacağı anlamına gelmez. Daha çok organizasyon değerlendirmesi ile ilgilenir yani özel alanlardaki zayıflıkların iyileştirilmesine odaklanır.

YAZILIM GELİŞTİRME MODELLERİ

TÜMLEŞİK YETENEK OLGUNLUK MODELİ (CMMI - CAPABİLİTY MATURİTY MODEL İNTEGRATİON)

CMMI (Capability Maturity Model Integration) uzun vadede organizasyonun iş performansının olgunluğunun artırmasına yardım etmektedir. CMMI, organizasyonların işlemlerinin yeteneğini ve olgunluğunu değerlendirmek için servis geliştirme, bakım, satın alma mekanizmaları için en iyi pratikleri sunmaktadır. Bu model tarafında içerilen geliştirme alanları, sistem mühendisliği, yazılım mühendisliği, tümleşik ürün ve süreç geliştirme, tedarikçi kaynakları ve satın alma alanlarıdır. CMMI, CMM'in üzerine geliştirilmiş olup sekiz yıldan beri kullanılmaktadır. CMMI geniş bir kullanım oranına sahiptir. Mart 2009'da Software Engineering Institute 3446 organizasyon ve 21141 projenin CMMI kullandığını açıklamıştır

YAZILIM GELİŞTİRME MODELLERİ

FEDERAL HAVACILIK YÖNETİM TÜMLEŞİK YETENEK OLGUNLUK MODELİ (FAA-İCMM FEDERAL AVİATİON İNTEGRATED MATURİTY MODEL)

FAA-iCMM federal havacılık yönetiminde yaygın olarak kullanılır. FAA-iCMM modeli, dış kaynak kullanımı ve kaynak yönetimini de içeren büyük yazılım sistemlerinde (enterprise) ilerlemeler yapmayı hedefleyen en iyi pratiklerden oluşan bir model sunar.

YAZILIM GELİŞTİRME MODELLERİ

GÜVENLİ CMM/GÜVENLİ YAZILIM METODOLOJİSİ (TRUSTED CMM/TRUSTED SOFTWARE METHODOLOGY (T-CMM, TSM)

90'lı yılların başında Strategic Defense Initiatives (SDI) "Güvenli Yazılım Geliştirme Metodolojisi" olarak adlandırılan bir model geliştirdi. Daha sonra bu model güvenli yazılım metodolojisi (Trusted Software Methodology (TSM)) olarak adlandırıldı. Bu model düşük seviyelerde bilmeden yapılan geliştirici hatalarına karşı yüksek seviyelerde ise bilerek yapılan zararlı yazılım ataklarına karşı direnç sağlayan seviyelerden oluşmaktadır. TSM daha sonra CMM ile birleştirilmiş (Harmonize) ve Trusted CMM ortaya çıkmıştır TCMM/TSM günümüzde yaygın olarak kullanılmamasına karşın ilerde yazılım geliştirme sürecinde bir kaynak olarak kullanılabilir.

YAZILIM GELİŞTİRME MODELLERİ

SİSTEM GÜVENLİK MÜHENDİSLİĞİ YETENEK OLGUNLUK MODELİ (SSE-CMMSYSTEMS SECURİTY ENGİNEERİNG CAPABİLİTY MATURİTY MODEL)

SSE-CMM bir organizasyonun güvenlik mühendisliği yeteneğinin değerlendirilmesi ve geliştirilmesi için kullanılabilecek bir süreç modelidir. SSE-CMM, güvenlik mühendislik pratiklerini genel olarak kabul edilmiş mühendislik prensiplerine göre değerlendirip kabul edilebilir bir çerçeve ortaya koyar. Böyle bir çerçeve güvenlik mühendislik prensiplerinin uygulamalarında performansı ölçme ve iyileştirmeyi sağlar.

YAZILIM GELİŞTİRME MODELLERİ

MİCROSOFT GÜVENLİ GELİŞTİRME DÖNGÜSÜ (SDL)

Microsoft SDL güvenli geliştirme döngüsü, Microsoft geliştiricilerinin daha güvenli yazılımlar geliştirebilme ihtiyaçları ve bu ihtiyaçlara yönelik arayışlarından ortaya çıkmış bir anaçatıdır. Temelleri Ocak 2002'de yayınlanan Trustworthy Computing (TwC) yönergesine dayanır. Bu model yazılım geliştirmenin başlangıcından itibaren güvenli yazılım geliştirme ve yaşam döngüsünü hedefler. Bu hedeflere ulaşmak için, eğitim ve farkındalık, proje başlangıcı, en iyi pratikleri tanımla ve uygula, risk analizi yap, risk analizi aracı, risk analizi, yazılım dokümantasyonu araçları ve müşteriler için en iyi patrikler, güvenli kodlama politikası, güvenli test politikası, güvenlik ekleme, son güvenlik kontrolü, güvenlik müdahale planlaması, ürün çıkarma güvenlik cevapları ve işletme olmak üzere on üç adımı kullanır

YAZILIM GELİŞTİRME MODELLERİ

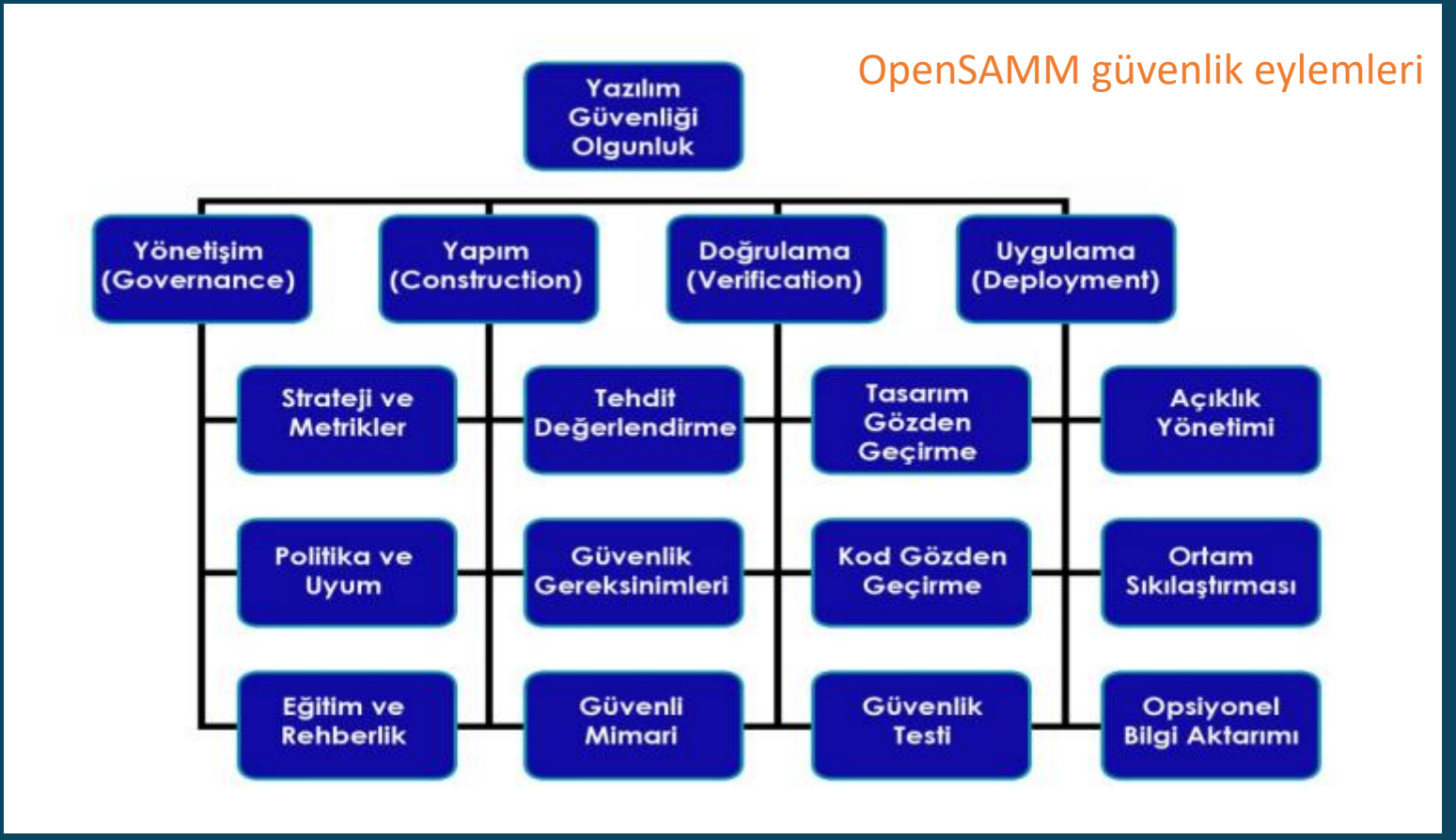
OpenSAMM MODELİ

OWASP (Open Web Application Security Platform) tarafından desteklenen OpenSAMM projesi kapsamında yazılım garanti olgunluk modeli (software assurance maturiy model) ortaya konmuşturu çalışmada güvenli yazılım geliştirme amacıyla bir ana çatı oluşturulmaya çalışılmıştır. Ana çatı, büyüklükten bağımsız bir şekilde her organizasyonun kendine adapte edebileceği, normal yazılım geliştirme döngülerine uyarlayabileceği ve bir organizasyondaki yazılım güvenliği alanında gelişmeyi yönlendirebilecek bir yapıda oluşturulmuştur. Anaçtı dört ana başlığa ayrılmıştır. Bu başlıklar, yönetişim (governance), yapım (construction), doğrulama (verification) ve uygulama (deployment) olarak belirlenmiştir. Bu ana başlıklar aslında normal yazılım geliştirme döngüsünün temel adımlarına karşılık gelmektedir. Her bir ana başlık altında üçer güvenlik eylemi yer almaktadır. Bu yapıda, güvenli bir yazılım geliştirmek için her bir temel yazılım geliştirme adımına karşılık yapılması gereken güvenlik çalışmaları güvenlik eylemi olarak değerlendirilmiştir. Her bir eylem altında da organizasyonun o eyleme konu alan alandaki olgunluk seviyesine göre hedefler (objectives) belirlenmiştir.

YAZILIM GELİŞTİRME MODELLERİ

OpenSAMM MODELİ

OpenSAMM güvenlik eylemleri



OpenSAMM MODELİ

Yönetişim başlığı altında, organizasyonda uygulanacak yazılım güvenliği programının stratejik yönünün belirlenmesi, organizasyona ait güvenlik çalışmalarının performansını ölçme yöntemlerinin ortaya konması, belirlenmiş kurum içi standartlara ve kurum dışı düzenlemelere uyum sağlanması ve çalışanların yazılım güvenliği konusunda eğitilmesi ile söz konusu çalışanlara rehberlik yapılması konuları ele alınmaktadır. Yapım başlığı, güvenli yazılım oluşturmak için yazılım gereksinimi ve tasarımı aşamalarında gerçekleştirilmesi gereken güvenlik eylemlerine değinmektedir. Yazılımın karşılaşacağı tehditlerin değerlendirilmesi, güvenlik ihtiyaçlarının belirlenmesi ve güvenli mimarinin oluşturulması “Yapım” başlığının altındaki güvenlik eylemleridir. Doğrulama başlığı, tasarım, yazılım kodlama ve yazılım testleri aşamasında gerçekleştirilmesi gereken güvenlik gözden geçirmelerini ve güvenlik testlerini kapsamaktadır. Tasarım gözden geçirmesi, kod analizi ve güvenlik testleri bu kapsamdaki güvenlik eylemleridir. Uygulama başlığı ise yazılımın canlı sisteme kurulması ve desteğinin verilmesi aşamalarını kapsamaktadır. İlgili güvenlik eylemleri, açıklık yönetimi, ortam sıkılaştırması ve operasyonel bilgi aktarımı olarak ele alınmıştır.

YAZILIM GELİŞTİRME SÜREÇ MODELİ

YGSM, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar yazılım geliştirme süreç modeli olarak tanımlanır. Yazılım işlevleri ve ihtiyaçlar sürekli olarak değiştiği ve geliştiği için bir döngü biçiminde düşünülür. Yazılım yaşam döngüleri tek yönlü ve doğrusal olarak düşünülmemelidir. Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.

YAZILIM GELİŞTİRME SÜREÇ MODELİ

TEMEL ADIMLAR

Analiz: Bir problemin çözümü olarak nitelediğimiz yazılımların ne yapacağını ve nasıl yapacağını belirlediğimiz, personel ve donanım ihtiyaçlarının çıkarıldığı, olurluk aşamasının yapıldığı, proje planının oluşturulduğu yani problemi tanımladığımız aşama planlama aşamasıdır. Yazdığınız kod ancak isteneni doğru bir biçimde yerine getiriyorsa başarılı bir yazılımdır. Bu nedenle öncelikle yazılımdan ne istendiğinin doğru bir biçimde tanımlanması gerekir. Analiz aşaması personel, donanım ve sistem gereksinimlerinin belirlenmesi, sistemin fizibilite çalışmasının yapılması, kullanıcıların gereksinimlerinin analizi, sistemin ne yapıp ne yapmayacağının kısıtlamalar göz önüne alınarak belirlenmesi, bu bilginin kullanıcılar tarafından doğrulanması ve proje planı oluşturulması adımlarından oluşur.

TEMEL ADIMLAR

Çözümleme: Yazılım işlev ve ihtiyaçlarının ayrıntılı olarak çıkarıldığı aşamadır. Mevcut sistemde var olan işler incelenir, temel sorunlar ortaya çıkarılır, yazılımın çözümleyebilecekleri vurgulanır.

Tasarım: Çözümleme aşaması sonucunda belirlenen gereksinimlere yanıt verecek yazılımın temel yapısının oluşturulduğu aşamadır. Yazılım tasarımı, bir bileşen veya sistemin nasıl gerçekleştirileceğini belirlemek için kullanılan teknikler, stratejiler, gösterimler ve desenlerle ilgilidir. Bu aşama yazılım bileşenleri arasındaki içsel ara yüzler, mimari tasarım, veri tasarımı, kullanıcı ara yüzü tasarımı, tasarım araçları ve tasarımın değerlendirilmesi alt süreçlerini de kapsamaktadır. Tasarım aşaması, yazılımın hem kullanıcı ara yüzünü hem de programın omurgasını ortaya koymaktadır. Yapılacak tasarım, yazılımın işlevsel gereksinimlere uygun olmasının yanı sıra kaynaklar, performans ve güvenlik gibi kavramları da göz önüne alınarak gerçekleştirilmelidir.

TEMEL ADIMLAR

Mantıksal Tasarım: Önerilen sistemin yapısı anlatılır.

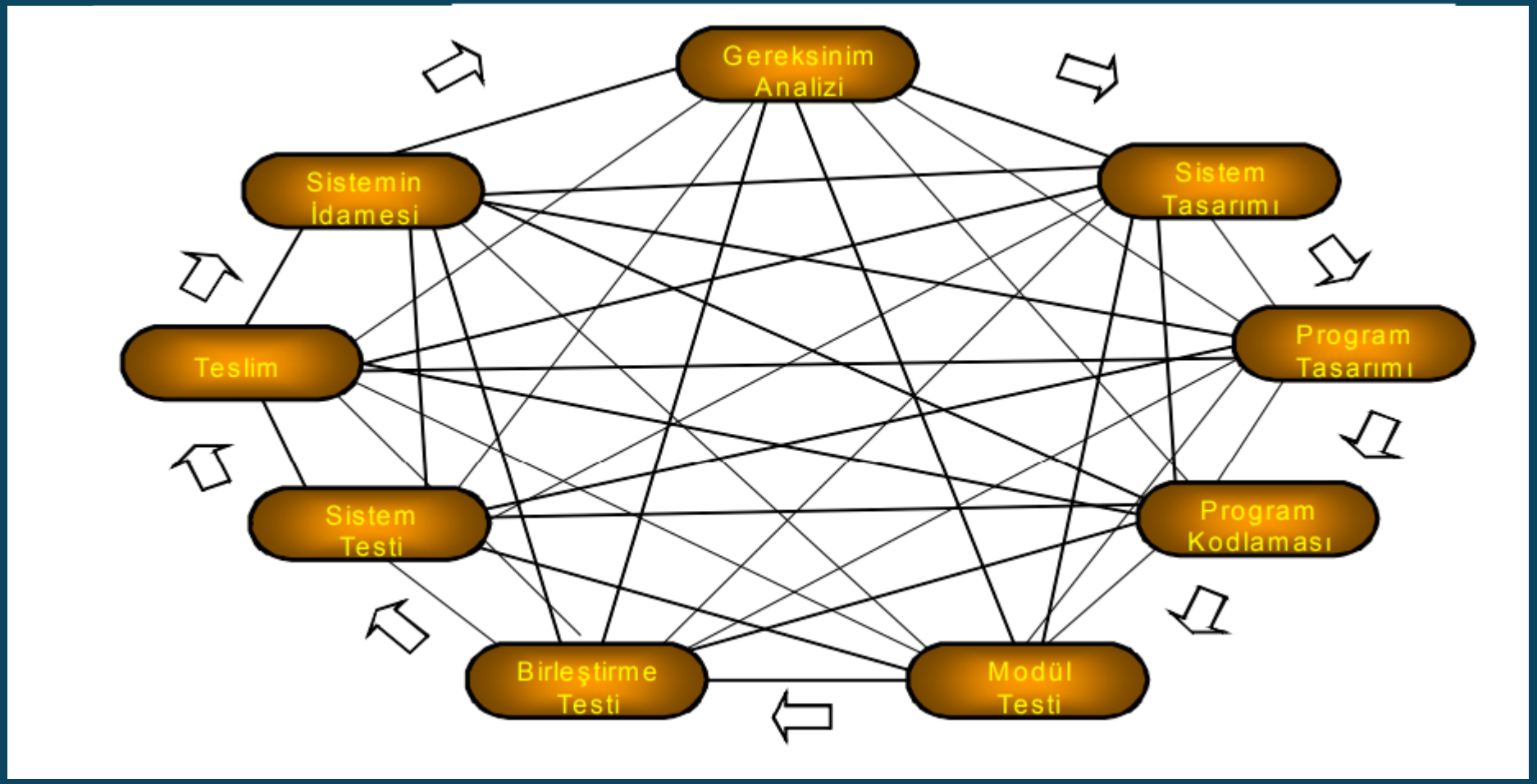
Fiziksel Tasarım: Yazılımı içeren bileşenler ve bunların ayrıntıları. (ekran tasarımları)

Gerçekleştirim: Kodlama, sınama ve kurma aşamalarının yapıldığı aşamadır. Kodlama aşaması, tasarım sürecinde ortaya konan veriler doğrultusunda yazılımın gerçekleştirilmesi aşamasıdır. Bu süreç programlama çalışmalarının yanı sıra yazılımın geliştirilmesi ve kullanıcıya ulaştırılması sürecindeki bütün çalışmaları kapsar. Tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışması olarak da nitelendirilebilir. yazılım geliştirme ortamı, programlama dili, veri tabanı yönetim sistemi, yazılım geliştirme araçları seçimi kodlama aşamasında gerçekleştirilir.

TEMEL ADIMLAR

Test: Test aşaması, yazılım kodlanması sürecinin ardından gerçekleştirilen sınama ve doğrulama aşamasıdır. Elde edilen uygulama yazılımının hem belirlenen gereksinimleri sağlayıp sağlamadığı hem de gerçekleştirimin beklentilere uygun olup olmadığını kontrol etmek için statik ve dinamik sınama tekniklerinden yararlanır. Statik teknikler, yazılımın tüm yaşam döngüsü boyunca elde edilen gösterimlerin analizi ve kontrolüyle ilgilenirken, dinamik teknikler sadece gerçekleştirilmiş sistemi içerir. Yazılım üretiminde ilk testler genelde geliştirme sürecinde programcı tarafından yapılır. Bununla birlikte, asıl hata ayıklama ve geribildirim hizmeti test ekipleri tarafından yapılır. Testler ve geribildirim müşteri yazılımı kullandığı sürece devam eder. Test sürecinde en faydalı geribildirimler son kullanıcı test gruplarından gelir.

Bakım: Yazılımın tesliminden sonra hata giderme ve yeni eklentiler yapma aşamasıdır. Yazılımın kullanıma başlanmasından sonra yazılımın desteklenmesi sürecini kapsar. Yazılımın eksiklerinin giderilmesi, iyileştirilmesi gibi alt aşamaları içeren aşamadır. Yazılım var olduğu sürece süre



Gerçek hayatta program geliştirme

YG SÜRECİNİN GENEL ÖZELLİKLERİ VE İÇİNDE YER ALAN ETKİNLİKLER

Genelde dayanılan model ne olursa olsun başarılı olabilmek için sürecin çatısını oluşturacak etkinlikler içinde aşağıdaki anahtar roldeki konuların nasıl karşılanacağının yanıtları bulunmalıdır:

- Yazılım geliştirme sürecinin denetimini sağlayacak proje yönetimi
- teknik olarak yöntemlerin uygulanış biçimi
- Her bir etkinliğin ara adımları
- Her bir etkinliğin ara ürünleri olarak (her aşamada yapılan tanımları sağlayan modellemeler, belgelemeler, veri tanımları, raporlar, formlar vb.) nelerin elde edileceği
- Kalitenin nasıl sağlanacağının yolu, yordamı
- Değişiklik isteklerinin nasıl derleneceğinin, nasıl ele alınacağının, nasıl yapılacağının, nasıl uygulama ortamlarında kullanılan yazılımlara aktarılacağının yöntemi

EN GENELDE YAZILIM GELİŞTİRME SÜRECİNİN 3 TEMEL AŞAMADA DÜŞÜNMEK GEREKİR:

Yazılımı Tanımlama Aşaması: Ne istendiğinin belirlenmesi üstünde odaklanılır.

- Yazılım Proje Yönetimi ve Gereksinim Çözümleme
- Hangi veriler işlenmeli
- Hangi bilgiler elde edilmeli
- Hangi işlemler nasıl yapılmalı
- Hangi işlevler görülmeli
- Nasıl bir davranış göstermeli
- Hangi ara yüzler kurulmalı
- Tasarımı kısıtlayıcı özellikler nelerdir
- Başarım hangi etkenlere bağlı olarak anlaşılacaktır

Yazılımı Geliştirme Aşaması: İstenenlerin nasıl sağlanabileceği üstünde odaklanılır.

- Yazılım Tasarımı, Kod Üretimi ve Yazılım Sınama
- Veriler nasıl düzenlenecek İşlemsel ayrıntılar nasıl yapılacak
- İşlevler hangi yazılım mimarisi içinde nasıl karşılanacak
- Kullanıcı ve mimari yapı içindeki kesimler arasındaki arayüzler nasıl olacak
- Tasarım hangi programlama dili/dilleri ile gerçekleştirilecek
- Her bir düzeyde sınamalar nasıl yapılacak

Yazılımı Geliştirme Aşaması: İstenenlerin nasıl sağlanabileceği üstünde odaklanılır.

- Yazılım Tasarımı, Kod Üretimi ve Yazılım Sınama
- Veriler nasıl düzenlenecek
- İşlemsel ayrıntılar nasıl yapılacak
- İşlevler hangi yazılım mimarisi içinde nasıl karşılanacak
- Kullanıcı ve mimari yapı içindeki kesimler arasındaki arayüzler nasıl olacak
- Tasarım hangi programlama dili/dilleri ile gerçekleştirilecek
- Her bir düzeyde sınamalar nasıl yapılacak

Yazılıma Bakım Desteğinin Verilmesi Aşaması: Teslim sonrası istenecek değişiklikler üstünde odaklanılır.

Yazılım Bakımı; düzeltici, uyarlayıcı, iyileştirici, önleyici

Düzeltici Bakım: En iyi yazılım kalite güvencesi sağlayıcı etkinlikler gösterilse de yazılımda belirlenememiş kusurlar uygulama başladığında ortaya çıkar. Düzeltici bakım işletime yeni girmiş yazılımda geç fark edilmiş kusurların giderilmesi için yapılır.

Uyarlayıcı Bakım: Zaman geçtikçe kullanılan bilgisayarların merkezi işlem birimi, işletim sistemi, iş görme kuralları, girdi / çıktı ortamları değişebilir. Yazılımların bu tür değişikliklere göre uyarlanması, üzerinde uygun değişikliklerin yapılması gerekir.

İyileştirici / Yetkinleştirici Bakım: Kullanıcı yazılımı kullandıkça yeni bir takım işlevler kazandırılmasını ya da bir işlevin farklı biçimde yerine getirilmesini isteyebilir. Geliştirici bakım ilk geliştirme sırasında olmayan gereksinimleri yazılıma kazandırmak için yapılır.

Önleyici Bakım: Yazılım üzerinde değişiklikler yapıldıkça tasarımla kazandırılmış yapısal niteliklerini yitirmeye başlar ve giderek kötüleşir. Bu yüzden yeniden mühendislik denilen bir çaba ile yapısının iyileştirilmesi gerekir. Bu yönde yapılan bakımla yazılımın tekrar kolayca değiştirilebilir, uyarlanabilir ve iyileştirilebilir bir biçime getirilmiş olur

YG SÜRECİNDEKİ ŞEMSİYE ETKİNLİKLER

Aşağıdaki etkinlikler Yazılım Geliştirmede önemli rolleri olan şemsiye etkinlikler olarak bilinir:

- Yazılım Projesi İzleme ve Denetim
- Biçimsel Gözden Geçirmeler
- Yazılım Kalite Güvencesi
- Yazılım Yapılandırma Yönetimi
- Yazılımın Belgelenmesi
- Yeniden Kullanılabilirlik Yönetimi
- Ölçümleme
- Risk Yönetimi

TEMEL ADIMLAR

Test: Test aşaması, yazılım kodlanması sürecinin ardından gerçekleştirilen sınama ve doğrulama aşamasıdır. Elde edilen uygulama yazılımının hem belirlenen gereksinimleri sağlayıp sağlamadığı hem de gerçekleştirimin beklentilere uygun olup olmadığını kontrol etmek için statik ve dinamik sınama tekniklerinden yararlanır. Statik teknikler, yazılımın tüm yaşam döngüsü boyunca elde edilen gösterimlerin analizi ve kontrolüyle ilgilenirken, dinamik teknikler sadece gerçekleştirilmiş sistemi içerir. Yazılım üretiminde ilk testler genelde geliştirme sürecinde programcı tarafından yapılır. Bununla birlikte, asıl hata ayıklama ve geribildirim hizmeti test ekipleri tarafından yapılır. Testler ve geribildirim müşteri yazılımı kullandığı sürece devam eder. Test sürecinde en faydalı geribildirimler son kullanıcı test gruplarından gelir.

Bakım: Yazılımın tesliminden sonra hata giderme ve yeni eklentiler yapma aşamasıdır. Yazılımın kullanıma başlanmasından sonra yazılımın desteklenmesi sürecini kapsar. Yazılımın eksiklerinin giderilmesi, iyileştirilmesi gibi alt aşamaları içeren aşamadır. Yazılım var olduğu sürece süre

YAZILIM GELİŞTİRME SÜREÇ MODELLERİ

Yazılım ile ilgilenen ve küçük-büyük projelerde yer alan-alacak olan herkesin bilmesi gereken modellerdir. Geçmişten günümüze kullanılmış olan modellerden kısaca bahsederek bunları bilmemizin bizlere ne gibi katkılar sağlayacağını, ne gibi getirileri olacağını daha iyi anlayabiliriz. Yazılım geliştirmek için kullanacağımız bu modeller planlı çalışmamızı ve geliştireceğimiz yazılımları en iyi biçimde geliştirmemizi sağlayacaktır.

GÜNÜMÜZ ‘YAZILIM GELİŞTİRME’ SÜRECİNDE HEDEFLENEN AMAÇLAR:

1) Üretim döngüsünde yazılım maliyetlerini azaltmak,
2) Üretilen yazılımın kalitesini arttırmak,
3) Üretici ve kullanıcı arasındaki iletişimi arttırarak istenilen seviyede ürün elde edilmesi

GÜNÜMÜZ 'YAZILIM GELİŞTİRME' SÜRECİNDE HEDEFLENEN AMAÇLARA Ulaşılabilmesi İçin Gerekli Koşullar Aşağıdaki Gibidir:

Prosedür: Ne yapılması gerekiyor? Burada yazılım geliştirme süreciyle ilgili nelerin yapılması gerektiği, bunların sonucunda sonuçların neler olacağı ve bu sonuçların detaylı analizinden sonra hangilerinin olmazsa olmaz (proje açısından kritik) olduğunun karar verilmesi aşamasıdır.

Metotlar: Amaca nasıl ulaşabiliriz? Burada belirlenmesi gereken ana hususlar, ilk seviye aktiviteleri, bunların sonuçları ve iş sahiplerine yapılacak sunuş yöntemleri şeklindedir.

Araç Gereksinimleri : Amaca ulaştıracak araçlar nelerdir? Burada değişik karakteristik özelliklerine göre hangi araçların yazılım geliştirme sürecinde kullanacağı ve bunların maliyet/performans analizlerinin tamamlanması aşamasıdır.

Bütün bu yukarıdaki aşamalar seviyesinde ana aktivite alanları ise:

A. Yazılım Geliştirme
B. Kalite Kontrol
C. Konfigürasyon Yönetimi
D. Proje Yönetimi

Yazılım geliştirilirken sürecin aşamaları ve adımlarından nasıl bir yaklaşım ile geçileceğini gösteren şimdiye kadar kullanılmış üreysel nitelikteki süreç modelleri şunlardır:

1. V Süreç Modeli,
2. Gelişigüzel Model,
3. Barok Modeli,
4. Araştırma Tabanlı Model,
5. Helezonik (Sarmal) Model,
6. Prototip Modeli,
7. Hızlı Uygulama Geliştirme Modeli,
8. Çağlayan (Şelale) Modeli,
9. Artırımsal Model,
10. Evrimsel Model
11. Yeniden Kullanılabilir Yazılım Modeli

V süreç modeli
(V process model)

Yazılım geliştirme süreç modelleri açısından uygulanabilirliği en kolay olan ve gerektirdiği maliyetler açısından en uygun olan model V-Modeli'dir. V-Modeli yazılım geliştirme sürecini koordine eden bütünleşik ve birleştirici methodların toplamıdır. Ana aktivite alanlarının birbirleri arasında koordinasyonunu sağlayan, tüm aktiviteleri içerik bakımından tanımlayan, proje kapsamında üretilecek tüm yazılım hakkında dokümantasyon bilgileri içeren ve bu dokümantasyonlar hakkında detaylı açıklama getiren ve son olarak da her seviye arasında problemin tanımlandığı andan ürün sürümüne kadar geçen tüm aşamalarda içsel kontrollü bir metodolojiler toplamıdır.

V süreç modeli
(V process model)

Bu modelde adından da anlaşılacağı gibi "V" yapısında bir yol izlenir ve adımlar bu şekilde gerçekleştirilir. Bu yol üzerinde sol taraf üretimi sağ taraf ise test işlemini ifade eder. Bu modelde yer alan çıktıları "Kullanıcı Modeli", "Mimari Model" ve "Gerçekleştirim Modeli" adı altında toplayabiliriz. Kullanıcı modelinde geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır. Mimari modelde sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler ele alınmaktadır. Gerçekleştirim modelinde de yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar ele alınmaktadır. Bu model belirsizliklerin az iş tanımlarının belirgin olduğu bilişim teknolojileri projeleri için uygun bir modeldir. Ayrıca model kullanıcının projeye katkısını artırmaktadır.

V-Modelin avantajları

1) Sonuç ürünün iş sahipleri tarafından proje başında arzu edilen ürün olması
2) İçerdiği kaliteli teknik dokümantasyon sayesinde kişilerden bağımsız olması. (Bu sistem ile yazılım geliştirici mühendislerin herhangi bir sebepten dolayı gerek proje üretim aşamasında gerekse sürüm sonrası ayrılmaları durumunda mevcut dokümantasyon ile yeni başlayan mühendislerin hızlı adaptasyonu ile sağlanan kişilerden bağımsızlık)
3) Paralel yürüyebilecek işlerin aynı zamanlamalarla yapılması sayesinde işin kolektif metotlara göre çok daha hızlı sonuca ulaşması.
4) Proje sürecinin herhangi bir yerinde iş sahibi tarafından yeni gereksinim istenmesi durumunda bunların projeye kolayca entegrasyonunun sağlanması.
5) Geliştirme sırasında ortaya çıkabilecek sorun ve hususların çok daha önceden belirlenebilmesi.
6) Standartlara bağlı geliştirme süreci sayesinde programın uygulanabilirliğinin yüksek ve modüler yapıda olması.
7) Sorunsuz bilgi aktarımı sayesinde kolayca uygulanabilen Risk Yönetimi.
8) İlerideki düşünülebilecek yeni modüllerin mevcut tasarıma uygun ve kolay adapte edilebilir şekilde entegrasyonuna izin vermesi.
9) Halen yürürlükte olan EURO-METHOD ile birebir uygunluk arz ettiğinden Avrupa Birliği standartlarına uygunluk.
10) Avrupa Birliği yüksek güvenlik seviyeli yazılımlar standartlarına uygunluk.
11) Bakım maliyetlerinin en az seviyeye çekilmesi.

YAZILIM UYGULAMA GELİŞTİRME PROJELERİNE ADAPTASYON

V-Model uygulamalarının en önemli özelliklerinden biri de herhangi büyüklükte ve ölçekteki bir uygulama yazılım projesine adapte edilebilir olma özelliğidir.

Bu süreçte takip edilmesi gereken aşamalar ise:

1)V-Modelinin belirlenmesi,

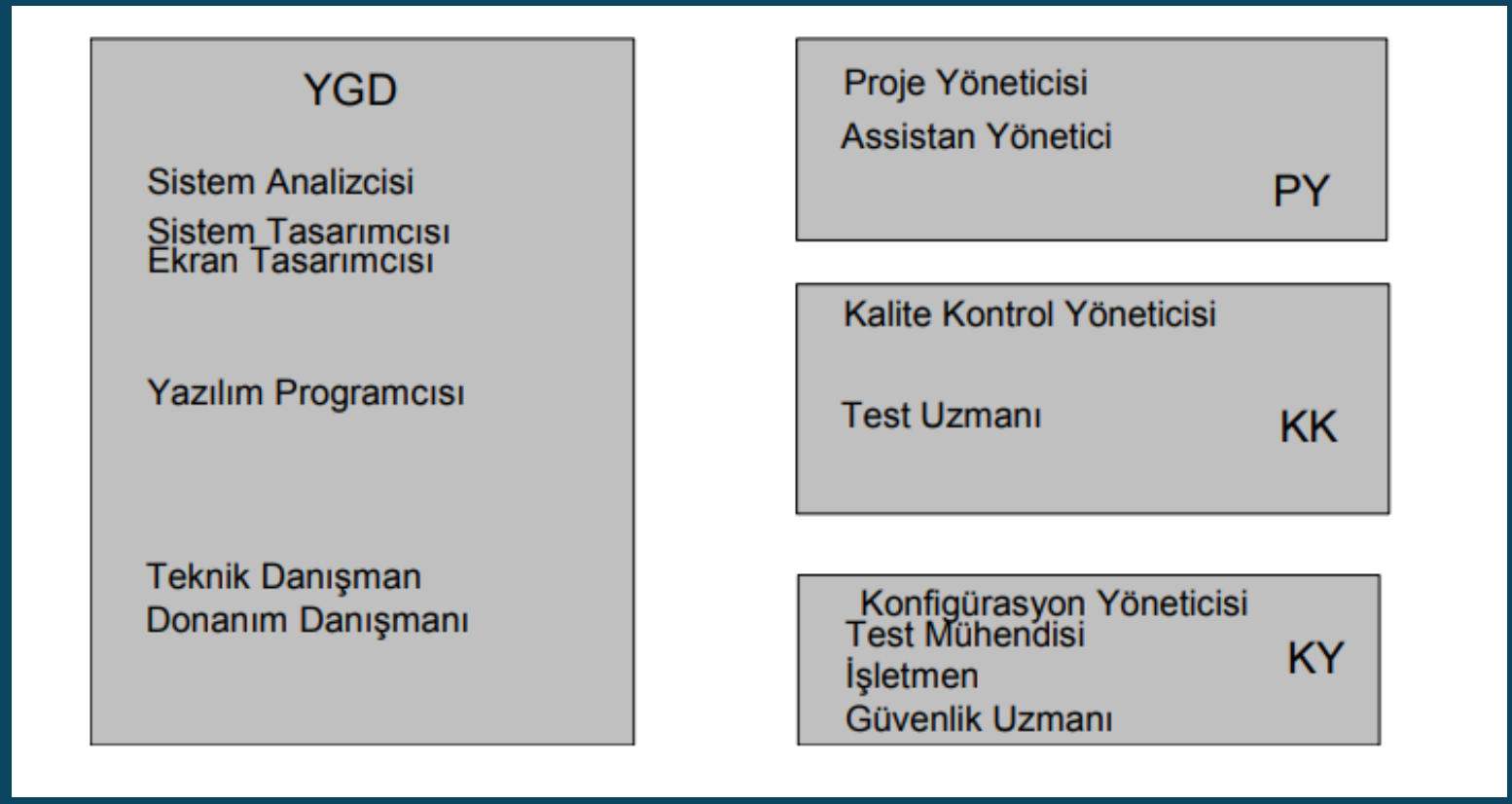
2) İş sahipleriyle çalışılarak model hakkındaki tavsiyelerin ve çıkarımların belirlenmesi,

3) Proje biçimlendirme çalışmaları sırasında: a. Gerekli aktivite ve ürünlerin seçilmesi, b. Proje için gereken diğer ayarlamaların yapılması, c. Proje rehberinin oluşturulması, d. Teknik çalışmalar sonucu değerlendirme ve maliyet analizi yapılması,

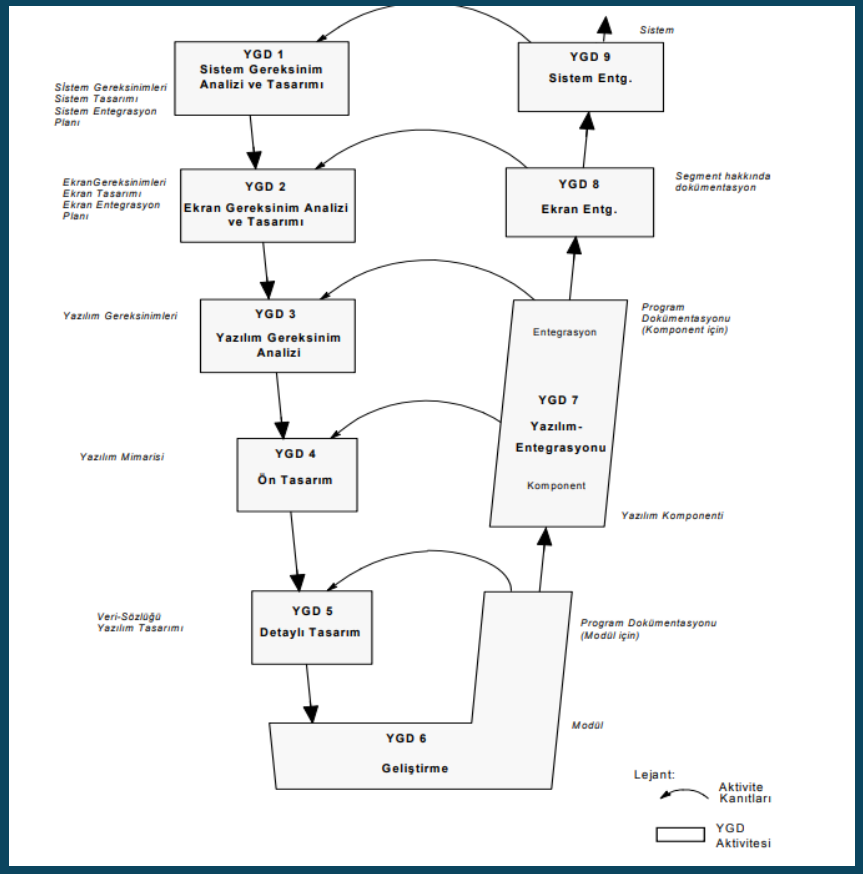
4) Proje Planının hazırlanması

V-MODEL DÂHİLİNDEKİ ROLLER VE TANIMLAR:

1) YGD: Yazılım Geliştirme Departmanı
2) PY: Proje Yönetimi
3) KK: Kalite Kontrol birimi
4) KY: Konfigürasyon ve Güvenlik Yönetimi



YAZILIM GELİŞTİRME SÜRECİNDE V-MODEL ŞEMASI



V-MODEL KULLANIM ALANLARI

1) E-Devlet uygulamaları
2) Askeri ve Savunma Sanayi Yazılım Projeleri
3) Finansal Yazılımlar
4) Anahtar Teslimi Yazılım Uygulama Projeleri
5) Avrupa Birliği yazılım geliştirme standardına uygun projelerde

GELİŞİGÜZEL MODEL (RANDOM MODEL)

Bu süreç modelinde herhangi bir yöntem yoktur. Yazılım tamamen geliştiren kişiye bağlıdır. İleri zamanlarda geliştirilen yazılımı hazırlayan kişi bile anlamakta zorluk çekebilir. Bu yüzden izlenebilirliği ve bakımı çok zordur. Daha çok 1960'lı yıllarda kullanılan bir yöntemdir. Genellikle tek başına yazılım geliştirenler tarafından tercih edilmiştir. Geliştirilen yazılımın programlaması diğer metotlarla geliştirilen yazılımların programlamasına göre basittir.

Barok modeli (Barok model)

Bu modelde yazılım yaşam döngüsünün temel adımları doğrusal bir biçimde takip edilir. Daha çok 1970'li yıllarda kullanılmış bir yöntemdir. Adımlar arası ilişkilerin tanımlı olmadığı bir yöntemdir. Günümüzde pek kullanılmayan bir yöntemdir. Bunun en büyük nedeni ise "Belgeleme" adımının bu modelde ayrı bir adım gibi ele alınıp yazılımın geliştirilmesi ve testinin ardından yapılmasıdır. Günümüzde kullanılan modellere aykırı bir durumdur. Günümüzde tercih edilen modellerde "Belgeleme" geliştirilen yazılımın ürünü olarak kabul edilmektedir. Ayrıca "Gerçekleştirme" adımını daha fazla ağırlık veren bir modeldir.

Barok Model'in adımları:

1. İnceleme
2. Analiz
3. Tasarım
4. Kodlama
5. Modül Testleri
6. Alt sistem Testleri
7. Sistem Testi
8. Belgeleme
9. Kurulum

Araştırma tabanlı model (Research-based model)

Bu model yap-at prototipi olarak da bilinir. Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır. Yapılacak işlerden edinilecek sonuçlar belirgin değildir. Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılır. Model-zaman-fiyat kestirimi olmadığı için sabit fiyat sözleşmelerinde uygun değildir. Bu model için örnek yazılım projeleri olarak şu örnekleri verebiliriz: en hızlı çalışan asal sayı test programı, en büyük asal sayıyı bulma programı, satranç programı gb.

Helezonik (Sarmal) model (Spiral model)

Artırımsal modelin daha denetimli yapılan ve sistematik bir biçim kazandırılmış halidir. Asıl amaç her bir artırımın güvenli ve hızlı yapılmasını sağlamaktır. Sarmal model kullanıldığında her bir artırımda yazılımın yeni bir sürümü ortaya çıkar. İlk artırımda ortaya çıkarılan kesim adeta bir prototipmiş gibi düşünülebilir. Prototipten tek farkı uygulama ortamında kullanılanın gerçek yazılım olmasıdır. Bir mühendislik uygulamasının gerektirdiği özelliklerde (kapasite, verimlilik, kalite gözetilerek) geliştirilmiştir. Sonraki yinelemelerde üstüne yeni işlevler yine bu yaklaşımla mühendisçe eklenecektir.

Helezonik (Sarmal) model (Spiral model)

Sarmalın her bir döngüsü belli etkinliklerden geçilerek tamamlanır:

• Müşteri ile yapılacaklara ilişkin iletişim; müşteri ile etkili iyi bir iletişimin kurulması yapılması gerekenlerin doğru belirlenmesi açısından zorunludur.

• Planlama; yapılacakların eldeki kaynaklar ve zaman çizelgesi gibi konular göz önünde tutularak planlanması.

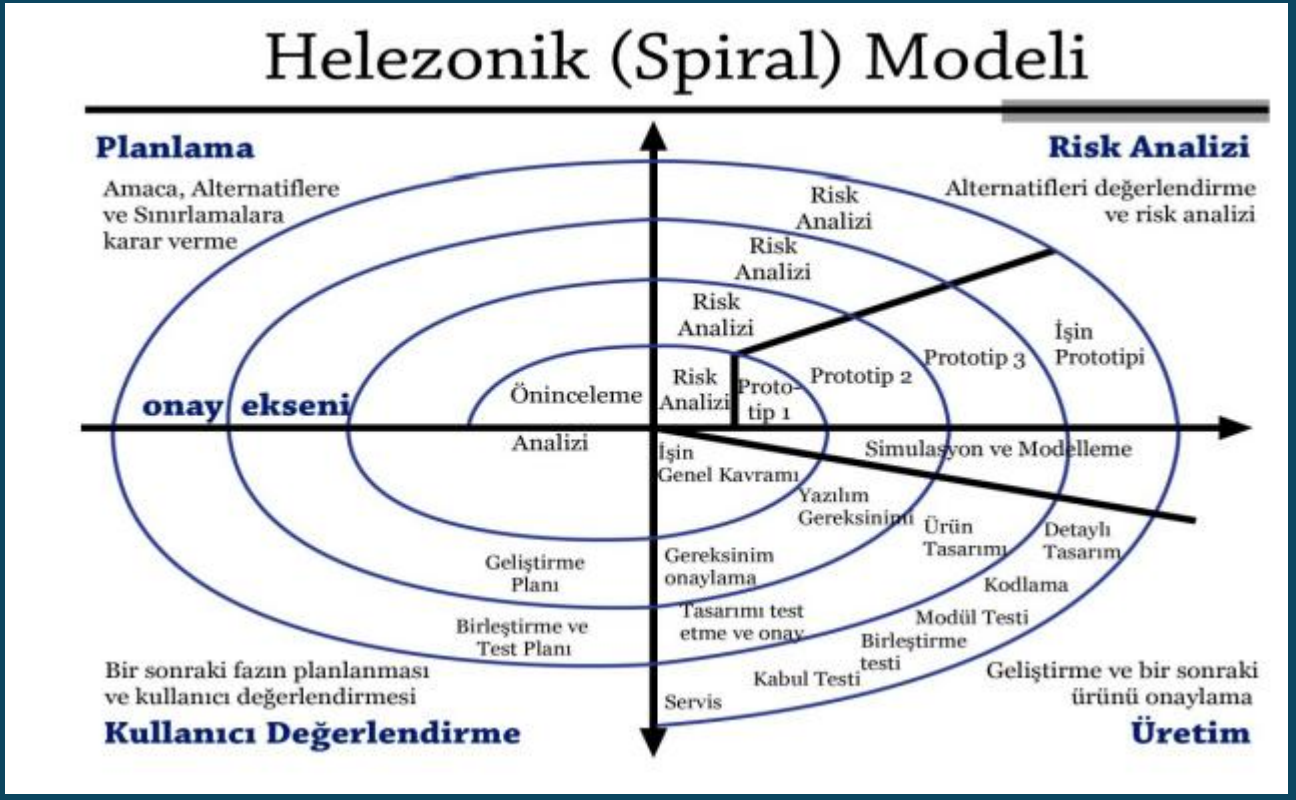
• Risk çözümleme; teknik ve yönetsel risklerin çözümlenmesi.

• Mühendislik işinin icrası; gereken işlerin yapılması, yazılımın gerçekleştirimi, çözümleme, tasarım, kodlama.

• Kurma ve sürüm; kurma, sınama, işletme ve kullanıcıya eğitim verilmesi.

• Müşterinin değerlendirmesi; kullanıcı geribildiriminin alınması, yapılanın geçerliliğinin ve gerektiği gibi çalıştığının ortaya konması

Helezonik (Sarmal) model (Spiral model)



Helezonik (Sarmal) model (Spiral model)

Sarmalın her bir döngüsü kendi içinde yapılması gereken yukarıdaki etkinliklere ayrılır. Etkinlikler projenin büyüklüğüne ve karmaşıklığına göre ayarlanabilir. Proje küçüldüğünde her bir etkinlikte yapılması gerekenlerin formalitesi azalabilir. Proje karmaşıklaştıkça da görevlerde yerine getirilmesi gerekenlerin biçimsel tanımı artırılarak dikkatli bir biçimde yerine getirilişleri izlenir. Ancak proje küçük ya da karmaşık olsun fark etmez; yazılım kalite güvencesi, risk yönetimi ve yazılım yapılandırma yönetimi gibi şemsiye etkinliklerin özenle yerine getirilmesi beklenir. Sarmalın her bir dönüşünde evrimleştirici etki artar. İlk döngü bazen yalnızca geliştirilecek yazılımın özelliklerini belirlemeye yönelik olabilir. İkinci döngü gerçekleştirim, üçüncü döngü iyileştirme döngüsü olabilir. Hatta araya gerek duyuluyorsa prototip geliştirme döngüsü sokulabilir. Sarmal model, yazılım bakım aşamasına da uygulanabilir. Yazılım geliştirme bittikten sonra bir sarmal döngü de bakım için geçilebilir. Böylece modelin tüm yazılım yaşam döngüsünü kapsaması sağlanabilir. Modele proje başlatma noktaları eklenebilir. Bu noktalar ilk eksen üstünde yer alır ve bir sonraki sarmal döngüyü başlatmak için karar noktaları olarak görülebilir Asıl olan projenin sürdürülmesi olmasına karşılık bu noktalar projeyi gerektiğinde durdurma / sonlandırma noktaları olarak da kullanılabilir.

Helezonik (Sarmal) model (Spiral model)

Sarmal model çok karmaşık büyük projelerde uygulanabilecek gerçekçi bir model olarak görülmektedir. Çünkü süreçte ilerledikçe her bir evrimleşmede geliştiriciler ve müşteri birbirini daha iyi anlamaya başlar ve riskleri paylaşmayı öğrenir, riskleri birlikte karşılamaya alışırlar. Sarmal modelde, eğer yer verilmişse prototip geliştirme döngüsü daha çok risk azaltma mekanizması olarak düşünülür. Bir yerde doğrusal sıralı modelin adım adım sistemli ve denetimli bir geliştirme yaklaşımı içinde Yazılım Mühendisliğinin kuralları ile kullanılmasını sağlar, böylece uygulamadaki gerçek durumların sisteme yansıması daha çok sağlanmış olur. Bir başka önemli özelliği risklerin önceden öngörülmesi ile risklerin sorun yaratmasına yol açılmadan fark edilip önlenmesi için hazırlıklı olmaya zorlamasıdır.

Sarmal modelin dezavantajları:

Bu model yazılım geliştirmenin her derdine deva olarak görülemez. Sarmal süreç modelinin, yazılım geliştirilirken karşılaşılan her sorunu ve zorluğu azaltabildiği ancak tamamen ortadan kaldırabildiği söylenemez. Özellikle böyle bir modele dayanarak yazılım geliştirme projesine girmeye müşteriyi razı etmek çok zordur. Özellikle de sözleşme yapılarak geliştirilecek projeler için uygun değildir. Müşterinin izlenecek yaklaşımın en çok denetim sağlayıcı bir süreç modeli olduğuna ikna edilebilmesi çok zordur. Projenin başarısı büyük ölçüde risk yönetimindeki başarıya bağlı olduğundan risk yönetiminde iyice uzmanlaşma gerektirmektedir.

Sarmal modelin avantajları:

1. Kullanıcı Katkısı - Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. - Yazılımı kullanacak personelin sürece erken katılması ileride oluşabilecek istenmeyen durumları engeller.
2. Yönetici Bakışı - Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır.
3. Yazılım Geliştirici (Mühendis) Bakışı - Yazılımın kodlanması ve sınanması daha erken başlar. - Risk Analizi Olgusu ön plana çıkmıştır. - Her döngü bir fazı ifade eder. - Doğrudan tanımlama, tasarım… Vs gibi bir faz yoktur. - Yinelemeli artımsal bir yaklaşım vardır. - Prototip yaklaşımı vardır

Prototip model
(Prototyping model)

Yazılım geliştirme projelerinde kullanıcıların genellikle gereksinimlerine ilişkin çok genel tanımları ancak yapabildikleri gözlenmiştir. Özellikle girdiler, işlemler ve çıktılar konusunda ayrıntılı tanımlar yapamazlar. Böyle olunca geliştiriciler uygulanacak algoritmaların işlevselliğinden, insan-makine etkileşiminin etkinliğinden, girdi ve çıktı biçimlerinin uygunluğundan emin olamazlar. Bu konularda belirsizliğin çok olduğu durumlarda prototip geliştirme modeli uygulanabilir ciddi bir seçenek oluşturmaktadır.

Prototip model (Prototyping model)

Prototip geliştirme modelinde, gereksinimler hızlıca toplanarak işe başlanılır. Geliştiriciler ve kullanıcılar aynı masa etrafında buluşarak yazılımdan elde edilecek bütün çıktılara, bu çıktılar için gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verirler. Daha sonra hızlıca yapılan bir tasarım ile yazılımın kullanıcıya yansıyacak yönünü aktaran bir prototip üretilir. Prototip kullanıcının kullanımına ve değerlendirilmesine sunulur. Bu değerlendirmelere bakılarak prototip üzerinde gerekli değişiklikler yapılır. Prototipin yeni hali kullanıcı tarafından yeniden değerlendirilir. Böylece kullanıcının istediği yazılıma iyice yaklaşılmış bir prototip üzerinde yazılımın neler yapacağı konusunda kullanıcı ile anlaşmaya varılır.

Hızlı uygulama geliştirme modeli (The RAD - Rapid application development model)

Kullanıcının belli gereksinimlerinin sağlanması için çok kısa süreler içinde ortaya çıkartılan kapsamı daraltılmış alt sistemlere ilişkin yazılımlardır. Bu model Doğrusal Sıralı Modelin çok hızlandırılmış bir uygulaması gibi görülebilir. Gerçekleştirimin hızlı yapılabilmesi genelde bileşen tabanlı olmasını gerektirir. Hazır bileşenler gereksinimleri karşılayabiliyorsa 60-90 gün içinde çalışan bir program ortaya çıkartılabilir.

Hızlı uygulama geliştirme modeli (The RAD - Rapid application development model)

Bu modelin aşamaları şunlardır:

İş Modelleme:

- Kısıtlı bir işin gerektirdiği bilgi akışının ne olduğu hızlıca ortaya çıkartılır.
- Veri akışı içinde ortaya çıkan bilgilerin neler olduğu saptanır ve hangi işlemler sonucu bu bilgilerin elde edildiği belirlenir.
- Yapılması gereken işlemlerin kimin tarafından nasıl yapıldığı belirlenir.
- Elde edilen bilgileri kimin kullanacağı ve ulaştırılacağı yere karar verilir.

Hızlı uygulama geliştirme modeli (The RAD - Rapid application development model)

Veri Modelleme: İş modelinin içinde akan bilgilerin veri nesneleri olarak nasıl tanımlanacağına, aralarındaki ilişkilerinin nasıl kurulacağına ve nasıl korunacağına karar verilir. Yazılım veri tabanındaki verilere nasıl erişileceğine, verilerin nasıl işleneceğine, verilerin nasıl korunacağına ve verilerin gereksinim duyulan yerlere hangi biçimde ulaştırılacağına karar verilir.

Prototip model (Prototyping model)

İşlem Modelleme: İş modelinde verinin bilgiye dönüşümünü sağlayacak dönüşümlerin nasıl sağlanacağı belirlenir. Uygulama Geliştirme: Bu model 4. kuşak dillerin kullanılmasını öngörür. 4. kuşak diller programlamayı, pek çok ayrıntısını kodlayıcının üstünden alarak çeşitli çizelgelerde seçeneklerin işaretlenmesine dönüştürmüştür. Dolayısıyla isteneni yapan programlar hızla oluşturulabilir.
Bu dillerin yapılan uygulamanın kendisine ilişkin yeniden kullanılabilir program birimlerini (veri tabanına ilişkin tanımları, işlemlere ilişkin yordamları, girdi /çıktı görüntüleri vb. gibi) oluşturma ve bunları kullandırma yetenekleri bulunmaktadır.

Prototip model (Prototyping model)

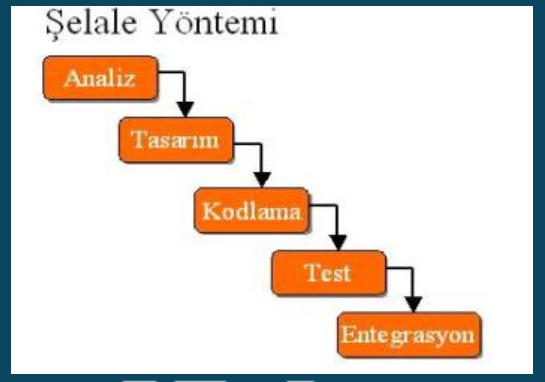
Sınama ve Başa Dönme: Yeniden kullanılabilirliğin sağlanması ölçüsünde sınama gereksinimi azalır. Çünkü yeniden kullanılan birimler daha önce sınanmış olacağından genelde bir daha sınanmalarına gerek kalmaz. Bu yüzden sınama zamanı da kısalır. Yalnızca yeni geliştirilmiş birimler ve ara yüzler sınanır.
Sakıncaları: - Kapsamlı büyük projelerde çok sayıda çalışma takımına gereksinim olacağı için pratik olmayabilir. - Hem geliştiricilerin hem de müşterinin hızlı çalışmaya bunun gereklerini yerine getirmeye kararlı olması gerekir. Her iki tarafın buna hazır olmaması durumunda başarı şansı yoktur. - Her tür uygulama için uygun değildir. Kendi doğası nedeniyle küçük modüllere bölünebilir olmayan uygulamalarda sorunlarla karşılaşılır. 45 - Performansın önemli olduğu uygulamalar için uygun değildir. - Teknik risklerin çok olduğu uygulamalar için uygun değildir. Eğer uygulama yeni bir teknolojiyi yoğun bir biçimde kullanacaksa ya da yeni geliştirilecek yazılım mevcut yazılımlar ile birlikte yoğun bir ilişki içinde ilişkilendirilerek işletilecekse riskler artar

Prototip model
(Prototyping model)

Yazılım geliştirme projelerinde kullanıcıların genellikle gereksinimlerine ilişkin çok genel tanımları ancak yapabildikleri gözlenmiştir. Özellikle girdiler, işlemler ve çıktılar konusunda ayrıntılı tanımlar yapamazlar. Böyle olunca geliştiriciler uygulanacak algoritmaların işlevselliğinden, insan-makine etkileşiminin etkinliğinden, girdi ve çıktı biçimlerinin uygunluğundan emin olamazlar. Bu konularda belirsizliğin çok olduğu durumlarda prototip geliştirme modeli uygulanabilir ciddi bir seçenek oluşturmaktadır.

Çağlayan (Şelale) modeli (Waterfall model)

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım gibi safhalardan oluşur. Geleneksel yazılım metotlarında bu safhalar şelale modelinde olduğu gibi lineer olarak işler. Her safha, başlangıç noktasında bir önceki safhanın ürettiklerini bulur. Kendi bünyesindeki değişikler doğrultusunda teslim aldıklarını bir sonraki safhanın kullanabileceği şekilde değiştirir.



Şelale modelinin özellikleri

- Şelalenin her basamağında yer alan aktiviteler eksiksiz olarak yerine getirilir. Bu bir sonraki basamağa geçmenin şartıdır.
- Her safhanın sonunda bir doküman oluşturulur. Bu yüzden şelale modeli doküman güdümlüdür.
- Yazılım süreci lineerdir, yani bir sonraki safhaya geçebilmek için bir önceki safhada yer alan aktivitelerin tamamlanmış olması gerekir.
- Kullanıcı katılımı başlangıç safhasında mümkündür. Kullanıcı gereksinimleri bu safhada tespit edilir ve detaylandırılır. Daha sonra gelen tasarım ve kodlama safhalarında müşteri ve kullanıcılar ile diyaloga girilmez

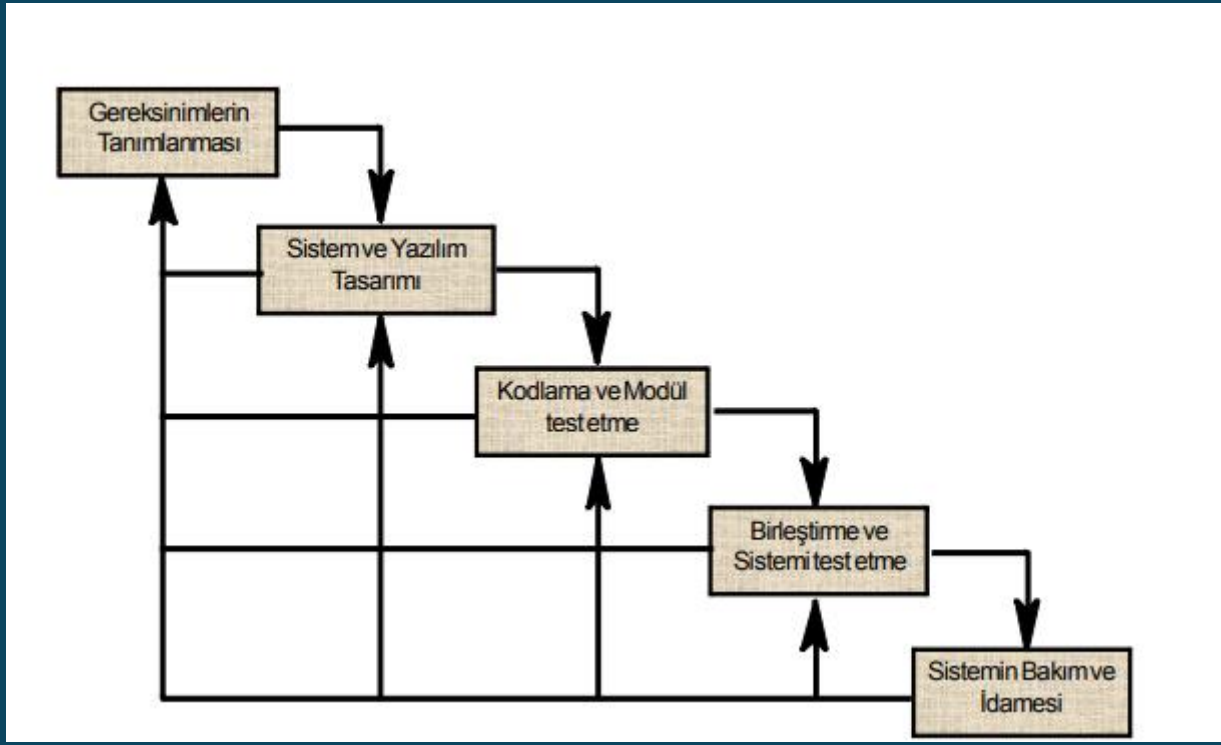
Şelale modelinin dezavantajları

Safhaların birbirinden kesin olarak ayrı tutulmaları gerçekçi değildir. Projelerde safhalar arasındaki bu sınırlar yok olabilir.

- Teoride safhalar birbirlerini takip edeler. Projelerde bunun bazen mümkün olmadığını ve önceki safhalara geri dönülmek zorunda kalındığını görebiliriz.
- Safhalar arası geri dönüş yetersizdir. Model değişikliğe açık değildir.
- Müşteri gereksinimlerinin proje öncesi detaylı olarak kâğıt üzerinde oluşturulması ilerde sorun yaratabilir. Müşteri gereksinimleri değişikliğe uğrayabileceği için, yazılım sisteminin de yapısal değişikliğe uğraması kaçınılmaz olabilir. Böyle bir durum maliyeti artırır, çünkü yeni ve değişen gereksinimleri implemente edebilmek için modelde yer alan safhaların birkaç kere uygulanması gerekebilir.

- Sistemin kullanılır hale gelmesi uzun zaman alabilir.
- Başlangıçta yapılan hataların tespiti çok uzun zaman alabilir. Bu hataların giderilmesi maliyeti yükseltir.
- Modül implementasyonları için zaman tahminleri proje planlarını oluşturan yöneticiler tarafından yapılır. Teknik bilgiye sahip olmayan şahıslar tarafından yapılan bu tahminler çoğu zaman doğru değildir. Bu durum proje planlama sürecini negatif etkiler

Şelale model işleyişi



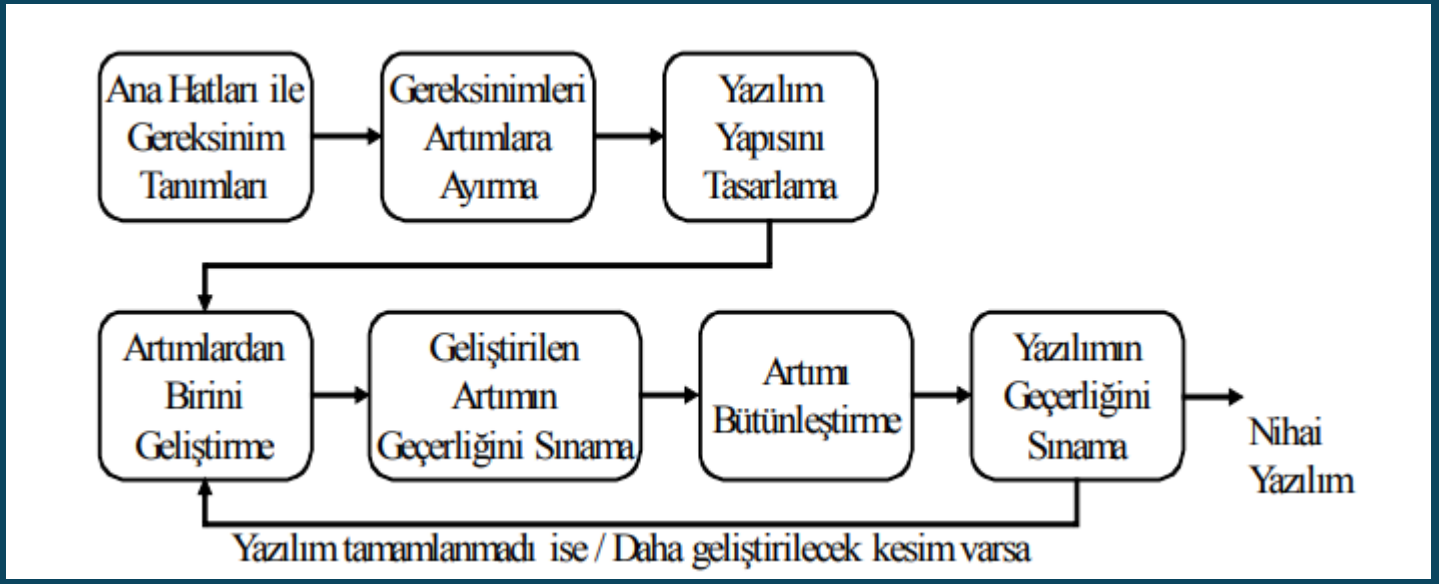
Proje başlangıcında her detayı göz önünde bulundurmak mümkün olmadığı için, şelale modeliyle geliştirilen yazılım sistemlerinin müşteri gereksinimlerini tam tatmin etmediğini görmekteyiz. Bunun önüne geçebilmek için projenin başlangıç safhasında analiz için çok zaman harcanır ve müşteri gereksinimleri en ince detayına kadar tespit edilir. Aslında proje başlangıcıyla oluşturulan dokümanlar obsolet (eskimiş) hale gelmiştir, çünkü müşteri gereksinimleri piyasa ve rekabet koşulları gereği değişikliğe uğramış olabilir. Ne yazık ki şelale modeli bunları dikkate almaz ve müşterinin talep ettiği değişiklikleri aza indirmeye çalışır. Bunun bir sebebi de sonradan gelen değişiklik taleplerinin maliyeti yükseltmesidir, çünkü bu durumda şelale modelinde yer alan safhaların birkaç kere uygulanması gerekebilir

Şelale modelinin avantajları

Yukarıdaki sorunlar yazılım geliştirme açısından iyice ciddiye alınması gereken durumlardır. Ancak doğrusal sıralı model, yazılım geliştirmenin nasıl yapılacağı konusunda tanımlı bir dayanak oluşturduğundan, bu modelle üstesinden gelinebileceği kestirilen projeler için uygulanabilir niteliktedir. Ne istendiğinin iyi bilindiği, gereksinimlerin proje başladığında eksiksiz ve tutarlı tanımlanabileceğine inanılan projeler için uygundur. Sadece gereklerin tam ve kesin olarak belirlendiği projelerde (ki böyle müşteri isteklerinin ve gereklerin başlangıçta sabitlenebildiği projeler pek görülmez) bu model kullanılabilmektedir.

Artırımsal model (Incremental model)

Doğrusal sıralı modele yinelemeli bir özellik katılarak artırımsal yazılım geliştirme modeli oluşturulmuştur. Artırımsal model bir takvime bağlı olarak yazılımı kesim kesim geliştirip teslim etmeye dayanır. Her bir yeni kesim öncekinin üstüne bazı ek işlevlerin eklenmesini öngörür.



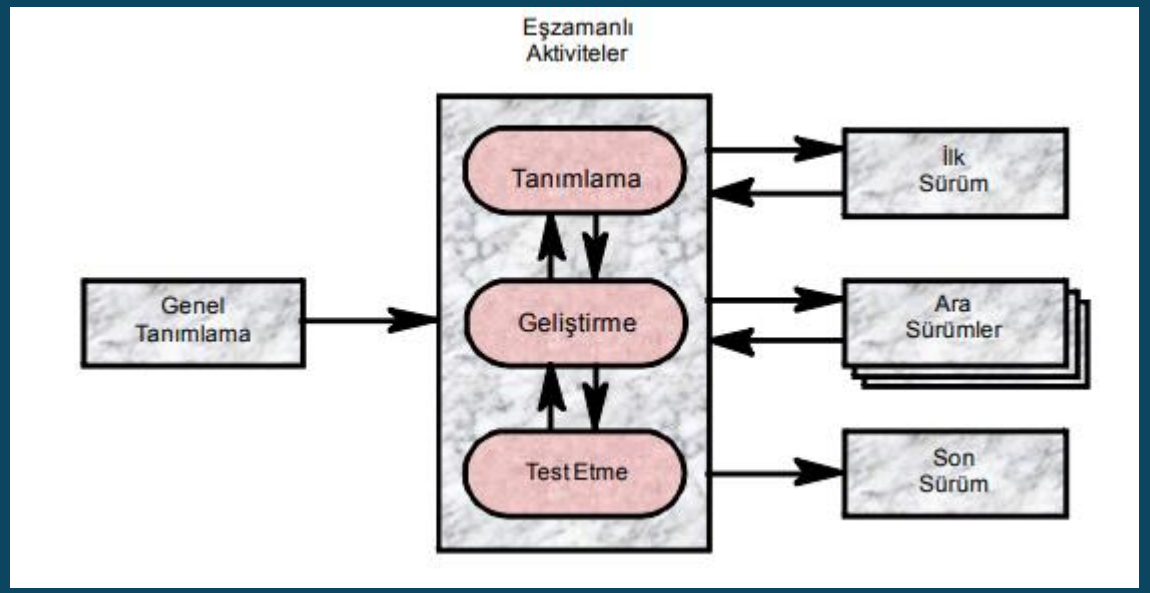
Artırımsal model uygulama şeması

Avantajları

Artımsal modelin en büyük avantajı, müşterinin süreç içerisinde daha fazla yer almasının sağlanabilmesidir. Kullanıcı, sistemin gereklerinin tek tek yerine getirildiğini gözlemleyebilmekte, varsa değişiklik önerisini hemen verebilmektedir.

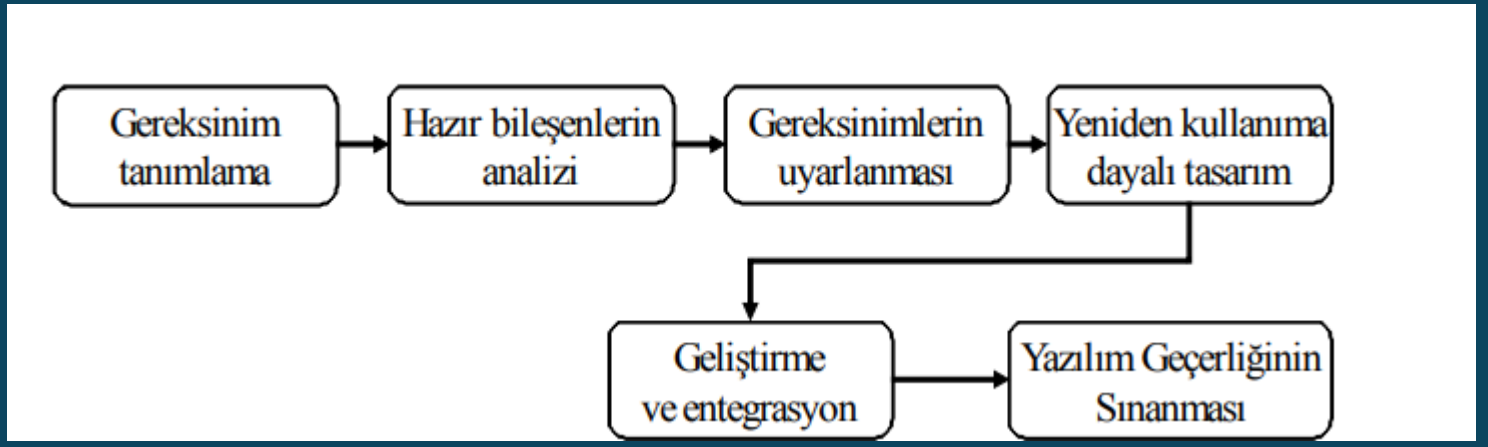
Evrimsel model (Evolutionary model)

İlk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir (banka uygulamaları). Her aşamada üretilen ürünler, üretildikleri alan için tam işlevselliği içermektedirler. Pilot uygulama kullan, test et, güncelle diğer birimlere taşı. Modelin başarısı ilk evrimin başarısına bağımlıdır.



Yeniden kullanılabilir model (Reusable model)

Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların (veya yazılım parçalarının) kullanılması ile geliştirme yapılması son yıllarda popülaritesi artan bir yaklaşımdır. Organizasyonların olgunlukları arttıkça, bu tür uygulamalar yapabilmek için alt yapı kurulmuş olmaktadır. Bu modelin işleyişi kısaca görülmektedir.



Yeniden kullanılabilir süreç model

Avantajlar:
Maliyetler azalıyor ve risk düşüyor. Yazılımlar daha hızlı gerçekleştiriliyor

Agile (Çevik) modelleme

Çevik modelleme (agile methods, yazılım sistemlerini etkili ve verimli bir şekilde modellemeye ve dokümantasyonunu yapmaya yönelik pratiğe dayalı yöntemlere verilen genel addır. Çevik modellemenin başlıca özelliği veri modelleri ve ara yüzü modelleri gibi modelleme tekniklerinin neler olduğunu ve bunların ayrıntılarını söylemek yerine bu tekniklerin nasıl uygulanması gerektiğini söylemesidir. Mesela; yapılan projelerin test edilmesi gerektiğini belirtse bile nasıl test hazırlanacağına değinmemesi gibi. O sadece bir yöntemler biçimidir ve bir projenin etkili, hızlı bir şekilde ortaya çıkarılmasında, müşterinin ihtiyaçlarını karşılamasında ve aynı zamanda da her türlü değişikliğe kolayca adapte olabilmesinde geliştiricilere yol gösterir.

Agile (Çevik) modelleme

Hangi durumlarda kullanılabilir?

Bu metotların kullanılmasının en uygun olduğu durumlar şunlardır:

- Projenin yazılım evresinde müşteriden gelebilecek talep değişikliklerinin tahmin edilemez olması
- Projenin parçalarının önce tasarlanıp ardından hemen geliştirilmesinin gerekmesi ve önceden ne yapılacağını, detaylı yol haritasını ve tasarımını tahmin etmenin çok güç olması
- Analiz, tasarım ve test etme süreçlerinin ne kadar zaman alacağının önceden bilinememesi
- Yazılım ekibinin birlikte çalışmak, hiyerarşiye önem vermemek, sağlam iletişim kurmak gibi özelliklere sahip olması

Çevik programlama metotlarının özellikleri

Çevik metotlar tam bir yazılım süreci değildir ama kapsamlı yazılım geliştirme yöntemlerini tamamlayıcı niteliktedir. Başlıca çevik süreç modelleri:

- Sınırsal programlama(Extreme Programming-XP)
- Çevik Birleştirilmiş Süreç (Agile Unified Process)
- Scrum
- Test Güdümlü Geliştirme (Test-driven Development)
- Çevik bilgi Metodu (Agile Data Method)
- Özellik güdümlü geliştirme (Feature-Driven Programming)

Agile (Çevik) modelleme

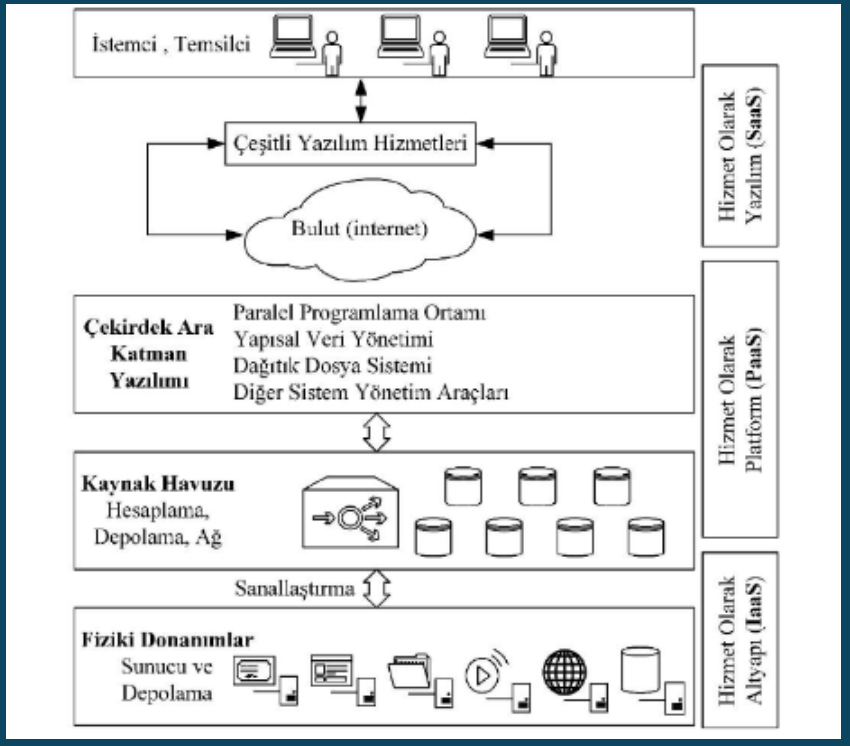
Çevik yazılım geliştirme, takım çalışmasını desteleyen liderlik vasıflarını, sorumluluk alabilme ve kendi kendini organize edebilme özelliklerini ve nitelikli ve yeterince hızlı yazılım geliştirmeye yarayan mühendislik uygulamalarını destekleyen proje yönetim sürecini kapsar. Bu yönetim süreci hem müşteri taleplerini hem de şirket amaçlarını düzenleyerek olumlu bir geliştirme ortamı ve yaklaşım sunar. Çevik modelleme işleri kısa vadeli planlar ve küçük gelişmeler halinde yapmayı uygun görür. Kısa vadeli planlar yineleme (iteration) olarak bilinir. Her yineleme sürecinde belli bir takım yazılım ya da modelle üzerinde çalışır ve bu süreç planlama, talep analizi, tasarım, kodlama, birim testi ve kabul testi gibi süreçleri kapsar. Yineleme süreci değişikliklere uyum sağlamayı kolaylaştırırken genel riski de azaltır. Tek bir yineleme bir ürünü piyasaya sürmek için ona yeterli işlevsellik katmayabilir ancak amaç her yineleme sonunda en az sorunla çalışan mevcut bir sürüm elde edebilmektir. Bir ürünü sürüm olarak piyasaya sürmek ve yeni özellikler eklemek için birden çok yineleme gerekir. Çevik programlama ve modellemede dokümantasyon da yapılır. Ancak dokümantasyon ürünün kullanılabilirliği ve verimliliğinin önüne geçmez. Aksine dokümantasyon sadece gerekli görülen yerlerde yapılır. IBM ’de çalışan ve aynı zamanda da “Effective Practices for Modeling and Documentation” kitabının yazarı olan scott ambler’e göre; geliştiricilerin tahtada yazıp çizdikleri modeller uzun uzadıya yazılan dokümanlardan daha kullanışlıdır ve insanların aylarca dokümantasyon yapmasının önlenmesi gerekmektedir

Bulut Sistem Altyapısı

Bulut sistem altyapısı, günümüzde teknolojik altyapıların temelini oluşturan bir konsepttir.
Bulut sistem altyapısı, veri ve uygulamaların internet üzerindeki uzaktaki sunuculara yerleştirilerek, kullanıcıların herhangi bir cihazdan, herhangi bir yerden erişebilmesini sağlar. Bu sistem, kaynakların paylaşılabilir olması, ölçeklenebilirlik, esneklik, güvenilirlik ve maliyet avantajları gibi birçok avantaj sunar.
Bulut sistem altyapısı, işletmeler ve bireyler için birçok fayda sunar. İşletmeler, kendi veri merkezi altyapılarını kurmak yerine bulut sistem altyapısını kullanarak, donanım, yazılım ve ağ kaynaklarına erişebilirler. Bu, işletmelerin kaynakları daha verimli kullanmasına, hızlı bir şekilde ölçeklenmesine ve yeni uygulamaları hızla hayata geçirmesine olanak tanır. Aynı zamanda, kullanıcılar farklı cihazlardan, herhangi bir yerden verilere ve uygulamalara erişebilir, çalışmalarını daha esnek ve verimli bir şekilde yönetebilirler. Bulut sistem altyapısı aynı zamanda veri güvenliği ve yedekleme gibi konuları da ele alır. Veriler, güvenli bir şekilde bulut sunucularında depolanır ve düzenli olarak yedeklenir, böylece veri kaybı riski en aza indirilir. Ayrıca, güvenlik protokolleri ve denetim mekanizmaları ile verilerin gizliliği ve bütünlüğü sağlanır. Bununla birlikte, veri merkezi altyapısının işletme maliyetleri yerine, bulut sistem altyapısı genellikle daha uygun maliyetlidir, çünkü kullanıcılar sadece kullandıkları kaynaklar için ödeme yaparlar, gereksiz kaynakları kullanmaktan kaçınabilirler ve ölçeklendirilebilir fiyatlandırma modelleri kullanabilirler

Bulut Hizmet Modelleri

Bulut hizmet modelleri; hizmet olarak altyapı (Infrastructure as a Service - IaaS), hizmet olarak platform (Platform as a Service - PaaS), hizmet olarak yazılım (Software as a Service - SaaS) olmak üzere üç kadegoriye ayrılır.



Bulut Hizmet Modelleri

Altyapı katmanı: Bu katman, hesaplama ve depolama kaynaklarını içerir.
Çerçevenin en alt katmanında, sunucular ve depolama gibi fiziksel cihazlar ve onanımlar, bir kaynak havuzu olarak sanallaştırılır ve işletim sistemi (OS) kurmak ve yazılım uygulamalarını çalıştırmak için kullanıcılara hesaplama, depolama ve ağ hizmetleri sağlamak için kullanılır. Bu nedenle hizmet olarak altyapı (IaaS) olarak adlandırılır. Tipik olarak, bu katmandaki hizmetler arasında Amazon'un Esnek Hesaplama Bulutu yer alır
Platform katmanı: **B**u katman, bulut bilişim sisteminin çekirdek katmanı olarak kabul edilir ve paralel programlama tasarımı ortamını, yapılandırılmış büyük veriler için dağıtılmış depolama ve yönetim sistemini, büyük veriler için dağıtılmış dosya sistemini ve bulut bilişim için diğer sistem yönetim araçlarını içerir. Program geliştiricileri, platform katmanının ana müşterileridir. Tüm platform kaynakları, program test etme, çalıştırma ve bakım gibi işlemler doğ-rudan platform tarafından sağlanır ancak son kullanıcılara değil. Bu nedenle, bir platform katmanındaki bu tür hizmetlere, hizmet olarak platform (PaaS) denir. Tipik hizmetler arasında Google App Engine (Cloud Computing Services | Google Cloud, 2023) ve Microsoft'un Azure'u (Bulut Bilişim Hizmetleri | Mic-rosoft Azure, 2023) yer alır.
Uygulama katmanı: Bu katman, basit yazılım ve uygulamaları ve müşteri arayüzlerini son kullanıcılara sağlar. Bu nedenle hizmet olarak uygulama (SaaS) adı verilir. Kullanıcılar, istemci yazılımını veya bir tarayıcıyı kullanarak İnternet üzerinden sağlayıcılardan hizmetleri çağırır ve su veya elektrik gibi yardımcı iş modeline göre maliyet öderler. İlk SaaS, Salesforce'tan Müşteri İlişkileri Yönetimi (CRM) idi ve force.com'a (Salesforce'ta bir PaaS) dayanarak geliştirildi. Google'ın belgeler, elektronik tablolar, sunumlar gibi çevrimiçi ofis
hizmetleri de tümü SaaS olarak adlandırılır

Bulut Dağıtım Modelleri

Genel Bulut (Public Cloud): Kullanıcılarına aynı donanım üzerinden hizmet sağlayan çok kullanıcılı bir modeldir. Kullanım maliyeti diğer bulut modellerine göre daha uygundur.

Özel Bulut (Private Cloud): Bulut sisteminin sadece tek bir kullanıcıya veya firma hizmet vermesi durumudur. Bu modeli tercih eden kullanıcılar bulut hizmetlerinin kullanımını kendi oluşturdukları veri merkezileri üzerinden kullanırlar. Bu durum fiziksel donanım gerektirdiğinden genel bulutla kıyaslanırsa daha maliyetlidir.

Hibrit Bulut (Hybrid Cloud): Genel ve özel bulut arasında karma bir sistem uygulanarak en verimli çalışma sistemini kullanıcıya sunan modeldir. Bulut modelleri arasında iletişimi sağlamak için VPN bağlantısı kullanılır. Bu durumu örneklendirirsek kullanılan uygulama genel bulutta barındırılsın, uygulama tarafından kullanılacak veriler şirket için önem araz ettiğinden şirete ait veri tabanının bulunduğu özel buluttan veriler çekilir

Topluluk Bulut (Community Cloud): Topluluk bulutu, belirli bir topluluğa sunulan bir hizmet türüdür. Bu hizmeti kullanan firmalar, paylaşımlı bir yapıda çalışır ve benzer özelliklere sahip diğer firmalar tarafından desteklenir. Topluluk bulutu, ortak iş gereksinimlerine sahip bir grup organizasyon tarafından kullanılan bir bulut çözümüdür. Bu topluluklar, özel gereksinimleri olan uygulamaları çalıştırmak için ihtiyaç duydukları ölçeklenebilirlik ve güvenliği sağlamak amacıyla bir araya gelirler. Bu hizmetler, belirli bir sektörde
faaliyet gösteren şirketler veya belirli bir bölgede bulunan işletmeler gibi benzer özelliklere sahip topluluklar için sunulabilir

Bulut Bilişim Sistemlerinin Avantajları ve Dezavantajları

Kişisel veya kurumsal anlamda yaşama veya iş hayatına kolaylık katan bulut bilişim sistemleri beraberinde getirdiği çözüm ve sorunları bulunmaktadır

BULUT BİLİŞİM SİSTEMLERİNİN AVANTAJLARI

Kaynakların Paylaşılabilir Olması: Bulut bilişim sistemi, kullanıcıların kaynakları (donanım, yazılım, ağ kaynakları vb.) paylaşabilir ve daha verimli bir şekilde kullanabilir. Bu, işletmelerin veya kullanıcıların ihtiyaçlarına göre kaynakları dinamik olarak ayarlayabilmelerini ve gereksiz kaynak kullanımını önleyebilmelerini sağlar. Ayrıca, birden çok kullanıcının aynı anda aynı kaynaklara erişebilmesi, iş birliği ve veri paylaşımını kolaylaştırır.

Ölçeklenebilirlik ve Esneklik: Bulut bilişim sistemi, kullanıcıların hızlı bir şekilde kaynakları ölçeklendirmesine ve esnek bir şekilde kullanmasına olanak tanır. Kullanıcılar, ihtiyaçlarına göre kaynakları artırabilir veya azaltabilir, uygulamalarını veya hizmetlerini genişletebilir veya daraltabilir, mevsimsel taleplere uyum sağlayabilir ve değişen iş gereksinimlerine hızlıca cevapverebilir. Bu, işletmelerin ve kullanıcıların hızlı ve etkili bir şekilde büyümelerini veya daralmalarını sağlar.

Bulut Bilişim Sistemlerinin Avantajları ve Dezavantajları

Güvenilirlik ve Veri Yedeklemesi: Bulut bilişim sistemi, kullanıcıların verilerini güvenli bir şekilde depolamalarını ve yedeklemelerini sağlar. Veriler, genellikle birden çok yerleşik yedekleme ve yedekleme mekanizmaları kullanılarak güvence altına alınır, böylece veri kaybı riski en aza indirilir. Ayrıca, güvenlik protokolleri, kimlik doğrulama ve yetkilendirme mekanizmaları gibi güvenlik önlemleri, kullanıcıların verilerinin gizliliğini ve bütünlüğünü sağlar.

Maliyet Avantajları: Bulut bilişim sistemi, kullanıcıların genellikle kullanılan kaynaklar için ödeme yapmalarını ve gereksiz kaynak kullanımından kaçınmalarını sağlar. Bu, kullanıcıların donanım veya yazılım satın almaları, işletmelerin kendi veri merkezleri veya altyapılarına yatırım yapmaları gibi maliyetli yatırımlardan kaçınmalarına yardımcı olur. Ayrıca, bulut bilişim sistemi, kullanıcıların abonelik veya kullanım temelli fiyatlandırma modelleri ile daha uygun maliyetli bir yaklaşım benimsemelerine olanak tanır

Bulut Bilişim Sistemlerinin Avantajları ve Dezavantajları

BULUT BİLİŞİM SİSTEMLERİNİN DEZAVANTAJLARI

Güvenlik Riskleri: Bulut bilişim sistemi, verilerin ve uygulamaların üçüncü taraf sağlayıcılara taşınmasını gerektirdiğinden, güvenlik riskleri taşımaktadır. Veri güvenliği, kimlik doğrulama, yetkilendirme ve veri bütünlüğü gibi konular, bulut sağlayıcının güvenlik önlemlerine ve politikalarına bağlıdır.
Güvenlik önlemlerinin yeterli olmaması, veri ihlalleri, veri kaybı veya kötü amaçlı saldırılar gibi risklere yol açabilir.

Bağımlılık: Bulut bilişim sistemi, kullanıcıların bulut sağlayıcısına bağımlı hâle gelmelerini gerektirir. Sağlayıcının hizmet kesintileri, teknik problemleri veya iflası gibi durumlar, kullanıcıların hizmetlere erişimlerini sınırlayabilir veya kesintiye uğratabilir. Bu yüzden iş sürekliliğinin kısıtlanması veya hizmet kesintisi riski oluşabilir.

Veri Gizliliği ve Uyumluluk: Bulut bilişim sistemi, verilerin üçüncü taraf sağlayıcılarda depolanması nedeniyle, veri gizliliği ve uyumluluk konularında endişeleri beraberinde getirebilir. Veri konumları, veri saklama politikaları, veri erişim yetkilendirmesi ve uyumluluk gereksinimleri gibi konular, kullanı-cıların dikkate alması gereken önemli faktörlerdir. Bazı sektörlerde belirli veri koruma ve gizlilik yasalarına tabi olabilir ve dolayısıyla bulut bilişim kullanımı kısıtlanabilir.

Bulut Bilişim Sistemlerinin Avantajları ve Dezavantajları

Hizmet Kalitesi ve Performans: Bulut bilişim sistemi, kullanıcıların hizmet kalitesi ve performansını doğrudan kontrol etme yeteneğini sınırlayabilir. Kullanıcılar, hizmet düzeyi anlaşmaları (Service Level Agreement - SLA) ile sağlayıcı tarafından sunulan hizmet seviyelerini garanti altına alabilirse de gerçek performans kullanıcı tarafından kontrol edilemez. Ayrıca, kullanıcıların internet bağlantılarındaki hız, güvenilirlik ve kullanılabilirlik gibi faktörlere bağımlı olmaları, hizmet kalitesini etkileyebilir.

Veri Transferi ve Bantgenişliği Maliyetleri: Bulut bilişim sistemi, büyük miktarda veri transferi gerektiren uygulamalar için veri transferi ve bantgenişliği maliyetlerini içerebilir.

Bulut Bilişim Sistemlerinde Veri Güvenliği

Bulut Sistemlerde bulunan verilerin korunması için birçok standart belirlenmiştir. Bu standartlar; ISO/IEC 27001 Bilgi Güvenliği Yönetim Standardı, ISO/IEC 27005 Bilgi Güvenliği Risk Yönetim Standardı, ISO 31000:2018 Risk
Yönetim Standardı şeklinde sayılabilir. 2014 yılında TSE çatısı bulunan Bulut Bilişimi ile ilgili olarak çalışmalar yapan Siber Güvenlik Özel Komitesi Bulut Bilişim Çalışma Grubunun düzenlediği Bilişim Güvenliği ve Kullanım Standardı taslağı oluşturulmuştur. Bu taslak ile işletmelere ait korunması gereken verilerin içerikleri belirlenmiş, oluşturulan çerçevede korunması gereken en önemli verilerin kişisel veriler olduğunun altı çizilmiştir

Güçlü Kimlik Doğrulama ve Erişim Kontrolü: Kullanıcıların güçlü şifreler kullanmalarını gerektiren kimlik doğrulama yöntemleri uygulayarak, yetkisiz erişimi önleyebilirsiniz. Ayrıca, kullanıcıların yalnızca ihtiyaç duydukları kaynaklara erişmelerini sağlamak için doğru erişim kontrollerini kullanmalısınız.
Veri Şifreleme: Bulut bilişim sistemindeki verilerin şifrelenmesi, veri güvenliğini sağlamak için önemlidir. Verilerin iletimi sırasında ve depolandığı sunucularda şifreleme kullanarak, verilerinizi koruyabilirsiniz.

Bulut Bilişim Sistemlerinde Veri Güvenliği

Güvenlik Duvarları ve Güvenlik Yazılımları: Bulut bilişim sisteminizi güvenlik duvarları ve güvenlik yazılımları kullanarak koruyabilirsiniz. Güvenlik duvarları, yetkisiz ağ trafiğini engelleyerek sistem güvenliğini artırabilir. Güvenlik yazılımları, kötü amaçlı yazılımları, virüsleri ve diğer tehditleri algılayarak engelleyebilir.

Veri Yedekleme ve Kurtarma: Bulut bilişim sisteminizdeki verileri düzenli olarak yedeklemek ve veri kaybı durumunda kurtarma planları oluşturmak, veri güvenliğini sağlamak için önemlidir. Veri yedekleme ve kurtarma stratejileri oluşturarak, veri kaybı durumunda verilerinizi geri yükleyebilirsiniz.

Hizmet Sağlayıcı Güvenliği: Bulut bilişim hizmet sağlayıcınızın güvenlik politikalarını ve uygulamalarını dikkatlice değerlendirmeniz önemlidir. Hizmet sağlayıcınızın güvenlik önlemleri, sertifikaları ve denetim süreçleri hakkında bilgi edinin ve güvenilir bir hizmet sağlayıcı seçin

Kullanıcı Eğitimi: Kullanıcıların bilinçli ve güvenli bir şekilde bulut bilişim sistemini kullanmalarını sağlamak için eğitim programları düzenleyebilirsiniz. Kullanıcılara güçlü şifreler kullanma, kimlik doğrulama yöntemlerini anlama, veri paylaşımı ve erişim kontrolü konusunda eğitim vererek, kullanıcı hatalarını ve güvenlik açıklarını azaltabilirsiniz.

Bulut Bilişim Sistemlerinde Veri Güvenliği

Sürekli İzleme ve Güncelleme: Bulut bilişim sistemlerinizdeki güvenlik önlemlerini sürekli olarak izlemeli ve güncellemelisiniz. Güvenlik tehditleri sürekli olarak değişmektedir, bu nedenle sistemlerinizi güncel tutarak güvenliği artırabilirsiniz. Yazılımları, işletim sistemlerini ve diğer bileşenleri düzenli olarak güncelleyerek, bilinen güvenlik açıklarını kapatmak önemlidir.

Veri Sınıflandırması ve Yetkilendirme: Hassas verilerinizi tanımlayarak ve sınıflandırarak, bu verilere sadece yetkili kullanıcıların erişimini sağlayabilirsiniz. Özellikle hassas verilerinizi yalnızca gerekli kullanıcılar arasında paylaşmalı ve gereksiz erişimi engellemelisiniz

Fiziksel Güvenlik: Bulut bilişim hizmet sağlayıcınızın veri merkezlerinin fiziksel güvenliği hakkında bilgi edinin. Fiziksel erişim kontrolü, güvenlik kameraları, yangın önleme sistemleri gibi fiziksel güvenlik önlemleri, veri güvenliğini sağlamak için önemlidir.

Güvenlik Olaylarını İzleme ve İnceme: Güvenlik olaylarını izlemek ve hızlı bir şekilde yanıt vermek, olası güvenlik ihlallerini önlemek için önemlidir. Olay günlüklerini izlemek, güvenlik tehditlerini tespit etmek ve önlemek için

Bulut Saldırıları

Bulut saldırıları, bulut sistemler üzerindeki servislere veya sunucu sistemlerine yapılan bir tür siber saldırıdır. Bu saldırı türünde, saldırganlar hedef sistemdeki kaynakları aşırı derecede kullanarak hizmetleri kullanılamaz hâle getirmeye çalışırlar. Bu kaynaklar arasında CPU, bellek, ağ bant genişliği ve disk alanı bulunabilir. Bulut saldırıları, hizmet kesintilerine neden olarak hem işletmelere hem de son kullanıcılara zarar verebilir. Bu nedenle, bulut hizmetleri kullanan şirketler, güvenliklerini sağlamak için doğru önlemleri almalı ve saldırıları önlemek için güçlü bir savunma stratejisi geliştirmelidir.

DoS Saldırı Modeli: Ağ sistemine bağlı bir sunucuyu meşgul ederek kullanıcıların sunucuya ulaşması önleyen bir saldırı modelidir. Bu saldırı modelinde hedef sunucuya ait veri akış trafiğini doldurmayı amaçlar. Bu sayede sunucuyu meşgul ederek sistemin gerçek kullanıcılara ait talepleri yerine getirmesi engellenir ya da tamamen sunucunun servis dışı kalması sağlanır

Hesap Ele Geçirme: Bulut Sistemine ait bir hesaba Brute Force saldırı tekniği ile şifreyi deneme yanılma yolu kullanılarak çözmeye yarayan bir saldırı tekniğidir.

Zararlı Yazılım (Malware) Saldırı Modeli: Solucan, fidye yazılımı, virüsler, truva Atları vb. gibi zararlı yazılımlarla bulut sistemine zarar verip sis-temin işleyişini bozma, kişisel verileri çalma, yok etme veya kendi amaçları için kullanmasına olanak sağlayan bir modeldir.

Bulut Saldırıları

İç Tehdit (Insider Threats) Modeli: Bulut sistemine girme yetkisi bulunan bir kullanıcını sisteme yasal olarak girerek yetkilerini bilerek veya yerek kötüye kullanması, veri sızdırılması, siber sabotaj gibi eylemlerle kurmun zarara uğramasını hedefleyen modeldir.
Yan Kanal (Side - Channel) Saldırısı: Sisteme fiziksel yollarla bağlanan cihazlara sızıp elde ettikleri verilerle sisteme dâhil olmayı hedefleyen saldırı modelidir. Bulut sisteminde bulunan bir kullanıcının bilgisayarına kötü amaçlı bir sanal makine yerleştirerek kullanıcıya ait parola ve kişisel bilgilere erişim
sağlanılabilir.
Çerez Zehirlenmesi (Cookie Poisoning) Modeli: Kullanıcının bilgisayarında depolanan verilerin bir web uygulaması tarafından izinsiz olarak değiştirilmesi durumudur. Tanımlama bilgisi kullanıcıların tarama geçmişleri ve tercihlerine ait verileri depolar.
Yanlış Güvenlik Yapılandırması: Buluta ait sistemlerin erişim kontrollerinin, sistem ve uygulamaların güvenli hâle getirilmeyip yanlış ayarlanması sonucu sistem açıklarının meydana gelmesi durumudur.
Güvenli Olmayan API: Saldırganlar tarafından sistemlere veya verilere erişmek, API'nin çalışmasını bozmak için güvenlik açıklarından faydalanabilirler
Bulut Kripto Madenciliği: Bu saldırı modeli ile bulut sistemine ulaşan saldırgan sistemi kripto madenciliği için kullanır. Kripto madenciliği; blockchain ağında bulunan işlemleri doğrulamak için karmaşık şekilde olan matematiksel problemleri çözme işlemidir.

Bulut Saldırılarına Karşı Korunma Yöntemleri

Güçlü Kimlik Doğrulama ve Yetkilendirme: Kullanıcıların güçlü ve karmaşık şifreler kullanmaları gerekmektedir. Çift faktörlü kimlik doğrulama kullanarak ek bir güvenlik katmanı ekleyebilirsiniz. Yetkilendirme politikala-rınızı düzenli olarak gözden geçirerek gereksiz kullanıcı erişimini engelleyebilirsiniz.

Veri Şifreleme: Verilerinizi hem dinlenen hem de depolanan aşamada şifrelemek, verilerinizi güvende tutmanın önemli bir yolu olarak kabul edilir.

Güvenlik Duvarları ve Güvenlik Grupları: Bulut bilişim hizmetlerinde güvenlik duvarları ve güvenlik grupları kullanarak ağınızı ve veri merkezinizi koruyabilirsiniz.

Sürekli İzleme ve Olay İzleme: Bulut ortamınızdaki olayları sürekli olarak izleyerek potansiyel güvenlik ihlallerini tespit etme ve hızlı yanıt verme yeteneğinizi artırabilirsiniz.

Veri Yedeği: Bulut bilişim hizmetleri kullanırken, verilerinizi düzenli ola-rak yedeklemek önemlidir

Güncel Yazılım ve Sistemler: Bulut hizmetlerinizde kullanılan yazılımarı, işletim sistemlerini ve diğer bileşenleri düzenli olarak güncelleyerek bilinen güvenlik açıklarını kapatmak önemlidir.

Bulut Saldırılarına Karşı Korunma Yöntemleri

Sağlayıcı Seçimi: Bulut bilişim sağlayıcısı seçerken güvenlik konusuna dikkat etmek önemlidir. Güvenilir ve güvenlik standartlarına uygun bir bulut sağlayıcısı seçerek, verilerinizin ve uygulamalarınızın güvenliğini artırabilirsiniz. Sağlayıcınızın güvenlik önlemlerini ve politikalarını inceleyin ve hizmet seviyesi anlaşmalarını (SLA'lar) dikkatlice değerlendirin

Yetkilendirilmiş Kullanıcı Erişimi: Sadece gerektiği kadar kullanıcılara erişim izni verin ve gereksiz kullanıcı hesaplarını kapatın. Her kullanıcının sadece kendi iş gereksinimleri için gerekli olan erişime sahip olmasını sağlayarak yetkilendirilmiş erişim uygulayın.

Sürekli Güvenlik Denetimleri: Bulut ortamınızdaki güvenlik denetimle-rini düzenli olarak gerçekleştirin ve güvenlik açıklarını tespit etmek için otomatik güvenlik denetim araçlarını kullanın. Potansiyel güvenlik açıklarını erken tespit ederek, gerekli düzeltici önlemleri alabilirsiniz.

Acil Durum ve Veri Kurtarma Planları: Bulut saldırılarına karşı acil durum ve veri kurtarma planları hazırlayın. Veri kaybı veya saldırı durumunda hızlı bir şekilde yanıt vermek için acil durum prosedürleri ve veri kurtarma planları oluşturun ve düzenli olarak gözden geçirin.

Zerarisk Tabanlı Güvenlik: Zero Trust (Zerarisk) tabanlı güvenlik yaklaşımını uygulayarak, kullanıcıları ve cihazları sürekli olarak doğrulayarak ve erişim kontrollerini sıkılaştırarak bulut ortamınızı daha güvenli hâle getirebi-lirsiniz. Kullanıcılar ve cihazlar, her erişim talebi için doğrulama adımlarından geçmelidir.