YMH 114 YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

Konular

- Ders içeriği, başarı değerlendirme
- YM kavramları, süreçler, modeller, yöntemler
- Yazılım Gereksinim Çözümlemesi
- Yazılım Tasarımı
- Yazılım Gerçekleştirimi
- Test ve Bakım
- Yazılım Geliştirme Yöntem Bilimleri
- Risk, Kalite ve Proje Yönetimi
- UML

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

Araştırma ve Proje

- Araştırma ve Sunum

(Gereksinim, Tasarım, Programlama (Kodlama/Hata Ayıklayıcı) (Kullanıcı Önyüz), Test, ProjeYönetimi, Düzenleşim (Konfigürasyon)/Değişim Yönetimi konularında kullanılan araç ve gereçleri araştır)

- Proje

(Bir konu üzerinde yazılım mühendisliği aşamalarının UML kullanılarak uygulanması)

Kullanılacak Yazılımlar

- Proje raporlama ve dökümantasyon işlemlerinde ihtiyaç duyulabilecek bazı araçlar şöyledir;
- MS Office Word, (proje raporu yazımı)
- MS Office Excel (Tablolama vb.)
- Adobe Photoshop (şekil çizimlerinde vb.)
- IBM RSA (Retional Software Architect) (UML diyagram çizimlerinde)
- MS Visio (Akış şeması çizimlerinde vb.)

Yazılım Nedir

Yazılım

– Tanımlanmış bir işlevi yerine getiren,
– Girdi ve Çıktıları olan,
– Herhangi bir donanım üzerinde çalışan,
– Bilgisayar programı veya programlarından ve
– Kullanım ve bakım kılavuzları gibi belgelerden oluşan

bir üründür.

Yazılım Mühendisliği (YM) - Nedir

"Mühendislik eylemlerinin, (Geliştirme, İşletme, ve Bakım), disiplinli, sistematik ve nicelikli bir şekilde yazılıma uygulanması"

YM – Önemi

- Yazılımın hayatımıza girmediği yer var mı?
- Yazılımsız hayat nasıl olurdu?
- Yazılım ve Eğitim
- Yazılım ve Ekonomi
- Yazılım ve Haberleşme
- Yazılımın verimliliğe katkısı
- Yazılımın kültüre etkisi

YM – Tarihçesi

- İlk Bilgisayarlar ve Makine ve Assembly Dili
- İşletim Sistemleri ve Anabilgisayarlar (Mainframe),
- Kart okuyucuları
- DOS ve PC'ler
- Derleyici (Compiler) ve Yorumlayıcılar (Interpreter) ve Yeni nesil yazılım dilleri
- Windows
- Yarı İletken teknolojisinin fiyat ve boyutlara etkisi
- Veri haberleşmesindeki gelişmeler
- **Internet**

Yazılım Geliştirme İstatistikleri

- Tipik yazılım projesinin geliştirilmesi 1-2 yıl sürüyor ve en azından 500.000 kod satır içeriyor
- Tüm projelerin yalnız %70-80'i başarıyla tamamlanıyor
- Tüm geliştirme sürecinde her birey günde ortalama 10 satırdan az kod yazıyor
- Geliştirme süresince her 1000 kaynak kod satırında 50-60 hata bulunuyor (satışa sunulmuş sistemde hata sayısı 4/1000'e düşüyor)

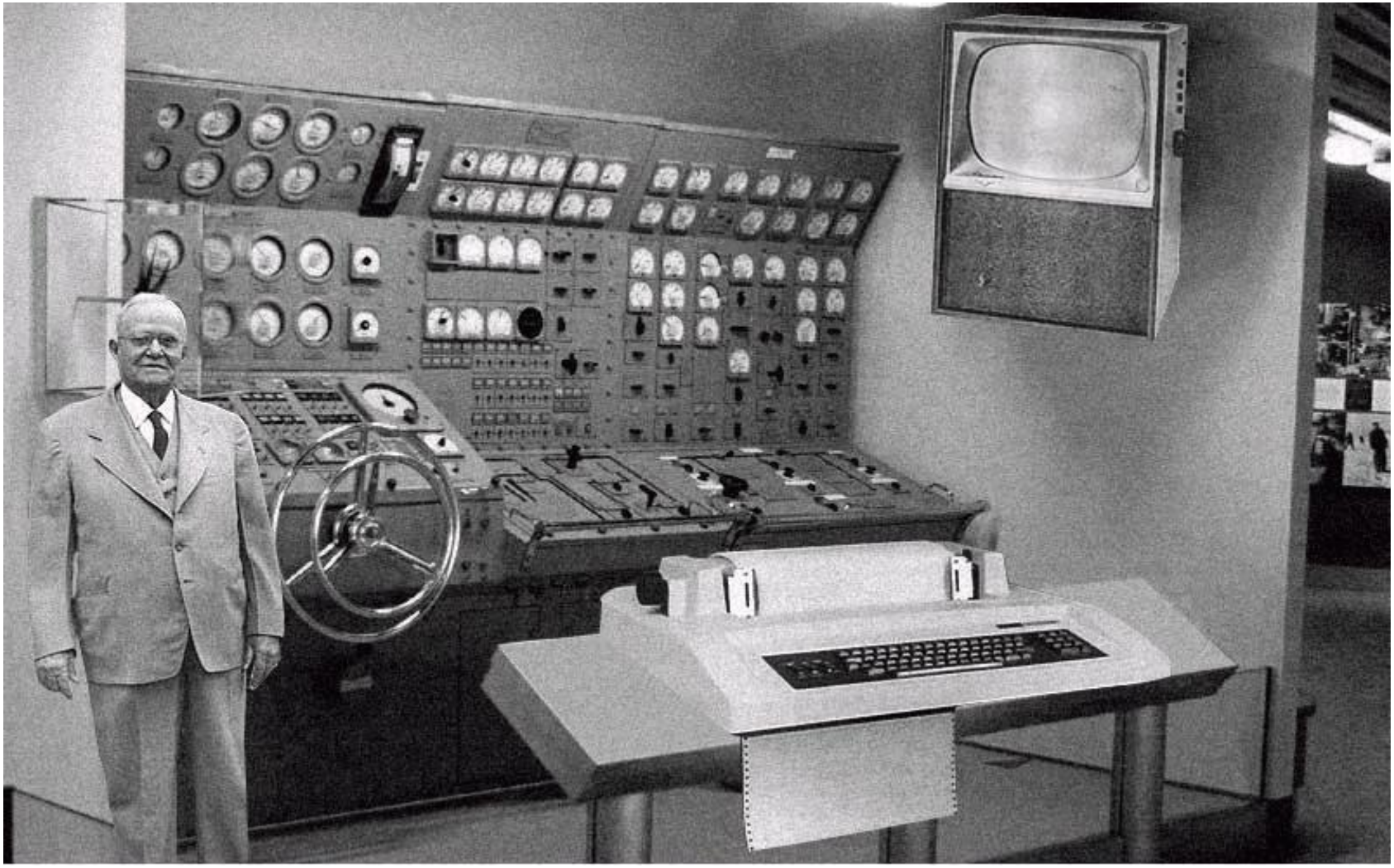
Yazılım Problemleri (Krizleri)

- Tasarlanan zamanın gerisinde kalma
- Bütçeyi aşma
- Düşük Kalite
  - Güvenilir olmayan yazılım
  - Kullanıcı taleplerinin karşılanmasında yetersizlik
  - Sürekliliğinin sağlanmasındaki zorluk

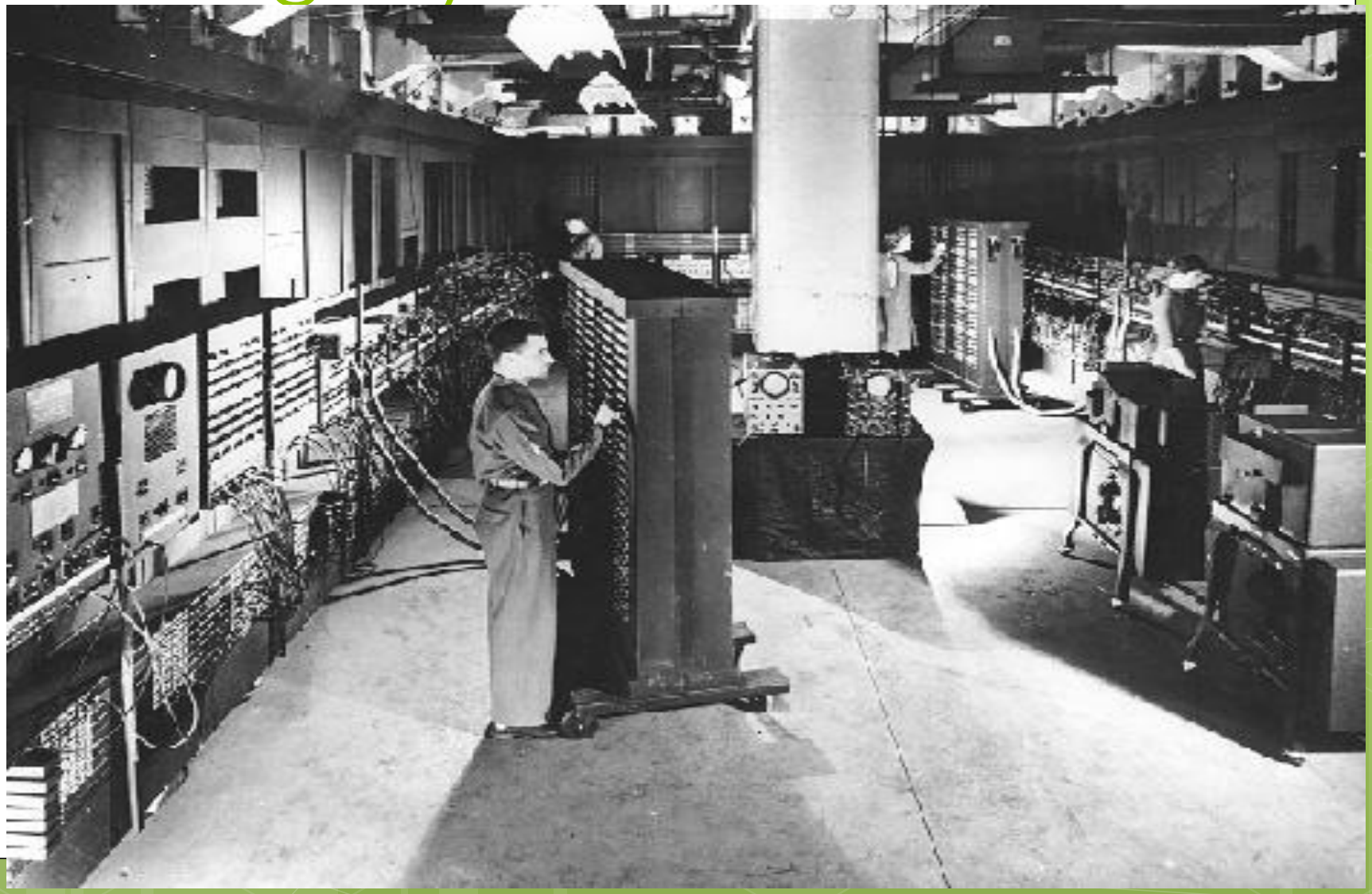
Yazılım-Donanım Evrimi

- Erken Yıllar 1950-1960 ilk bilgisayarlar, makine dili, 3-4 şirket
- Yazılımlar; kullanıcı ile birebir iletişimde bulunmayan, işlerin toplu olarak verilip, yalnızca yazıcı çıktılarının alındığı biçimde geliştirilmekte idi.
- Ayrıca, yazılımlar bu günkü gibi ürün tarzında değil, kuruluşa özel olarak geliştirilmekte idi.

İlk Bilgisayarlar



İlk Bilgisayarlar



Yazılım-Donanım Evrimi

- İkinci Dönem 1960-1975 Ana çatı, çok kullanıcılı sistemler,
  Veri tabanı yönetimi
- Çok kullanıcılı, gerçek zamanlı yazılımlar üretilmeye başlandı.
- VTYS önce yapılandırılmış dosya sistemleri ile ortaya çıktı.
- Yavaş yavaş ürün türü yazılımlar ortaya çıkmaya başladı.

Yazılım-Donanım Evrimi

- Üçüncü Dönem 1975-1990 Süper bilgisayarlar, Kişisel bilgisayarlar
- Açık sistem mimarisinin tanıtıldığı bu dönemde ürün bazlı yazılımlar oldukça yaygınlaştı.
- Kişisel bilgisayarlar yaygınlaşarak evlere girmeye başladı.
- Ağ yapısının gelişmesi ve güçlenmesiyle, dağıtılmış yazılım sistemleri geliştirilmeye başlandı.
- Yapay zeka teknolojisinin gelişmesiyle "akıllı uygulama yazılımları" üretilmeye başlandı

Yazılım-Donanım Evrimi

- Dördüncü Dönem 1990 - Yapay Zeka, Gömülü Sistemler, Paralel Sistemler, Yazılım Kaliteleri
- Uzman sistem yazılımları oldukça gelişmiş ve mikro-bilgisayarlar üzerinde yaygınlaşmıştır.
- "Yazılımda Kalite" olgusu önem kazanmış ve yazılım ile ilgili standartlar olgunlaşmaya başlamıştır.
- Yazılım üretimi ve ürünlerinin değerlendirilmesi amacıyla kurumlar oluşmaya başlamıştır.

Programlama Dillerinin Seviyeleri

1.Kuşak **Makine Dili**

10101110 10010001

2. Kuşak **Assembler**

8085, Z80, 68000, vs

3.Kuşak **Üst Seviye Diller**

Pascal Coral66 Basic

Bildirimsel

LISP Hope Prolog

Nesneye Yönelik Diller

Smalltalk, C++, Java

4. Kuşak **Veri Yapısal**

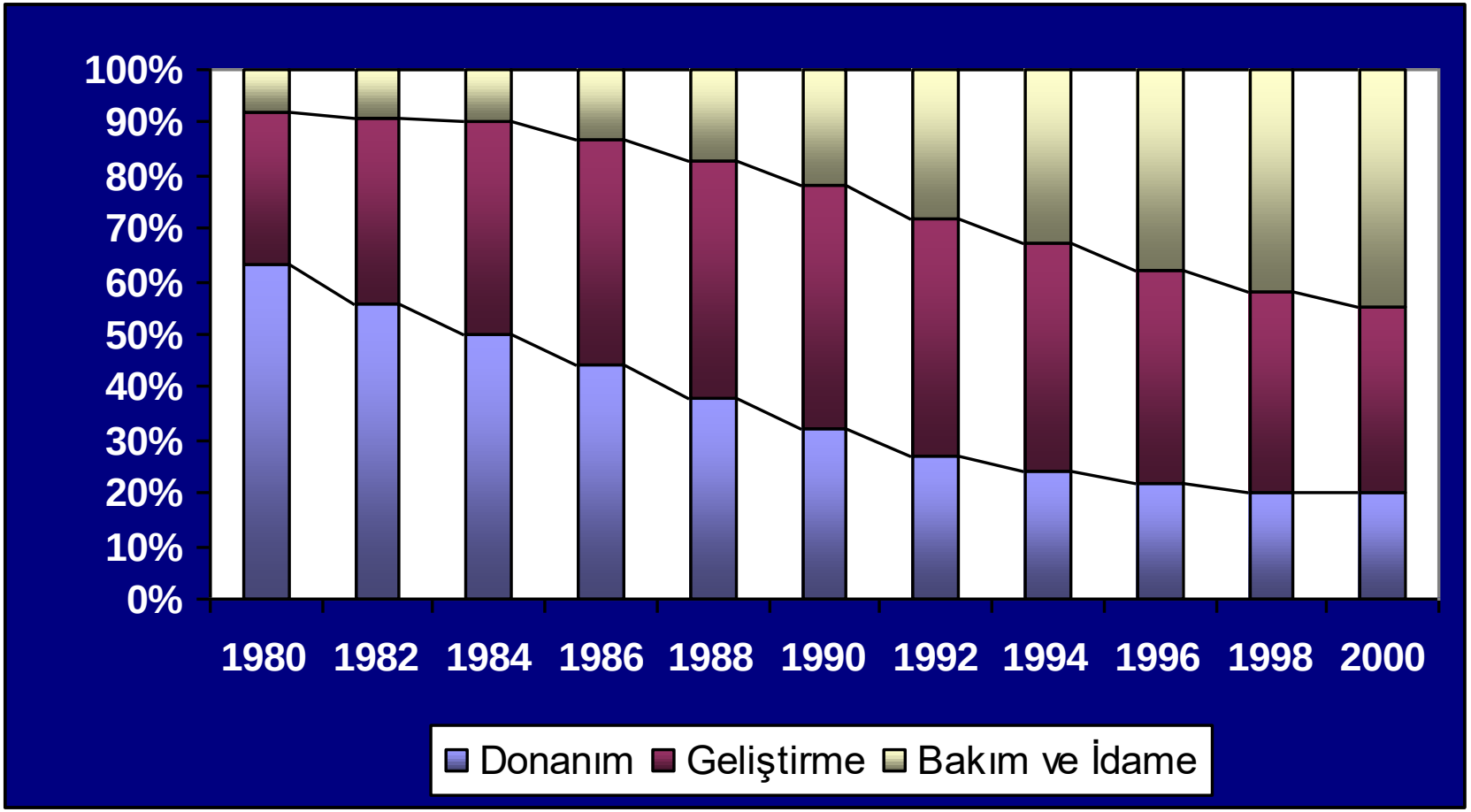
CICS, SQL

5. Kuşak **Yapay Zeka**

Ve Paralel Programlama

CSP, OCCAM

Sistem Harcamaları



Yazılım

Yazılım = Mantık (algoritma) +
Veri (test verisi, bilgi?) +
Belge (dokümanlar) +
İnsan (kullanıcı, geliştirici) +
Program (kod)

- "Bilgisayar sisteminin donanım bileşenleri dışında kalan her şey"

Yazılım

- Mantık, veri, belge, insan ve program bileşenlerinin belirli bir üretim amacına yönelik olarak bir araya getirilmesi, ve

  yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.

Mantık (algoritma)

- Bilgisayarlaştırılmak istenen işin mevcut mantığı yazılıma yansıtılmak durumundadır.
- Bu nedenle mantık (algoritma) bileşeni yazılımın en önemli bileşenlerinden biridir.

Veri

- Her tür yazılım mutlaka bir veri üzerinde çalışmak durumundadır.
- Veri dış ortamdan alınabileceği gibi, yazılım içerisinde de üretilebilir.
- Yazılımın temel amacı “veri”yi “bilgi”ye dönüştürmektir.

Belge (dokümanlar)

- Yazılım üretimi bir mühendislik disiplini gerektirir.
- Mühendislik çalışmalarında izlenen yol ya da kullanılan yaklaşımlar yazılım üretimi için de geçerlidir.
- Yazılım üretimi sırasında, bir çok aşamada yapılan ara üretimlere ait bilgiler (planlama, analiz, tasarım, gerçekleştirim, vb. bilgileri) belli bir düzende belgelenmelidirler.

İnsan (kullanıcı & geliştirici)

- İki boyutludur; yazılımı geliştirenler ve kullananlar.
- Günümüzde artık tek kişi ile yazılım geliştirmekten söz edilmemektedir.
- Yazılım üretimi için bir takım oluşturulmakta ve takımın uyumlu çalışabilmesi için çeşitli yöntemler geliştirilmektedir.

Program (kod)

- Yazılımın ana çıktısı sonuçta bir bilgisayar programıdır.
- Program işletime alındıktan sonra bakım çalışmaları sürekli olarak gündeme gelir.
- Bunun iki temel nedeni:
  - hiç bir program bütünüyle her olasılık göz önüne alınarak test edilemez.
  - işletmeler doğaları gereği dinamik bir yapıya sahiptir ve zaman içerisinde sürekli olarak yeni istek ve gereksinimler ortaya çıkabilmektedir.

Yazılım vs Donanım

- Yazılım geliştirilir vs donanım üretilir. (fabrika ortamında seri üretim)

- Donanım bileşenleri dışarıdan temin edilebilir, ancak yazılımı oluşturan parçalar için bu çoğu zaman mümkün değildir (günümüzde "yeniden kullanılabilir yazılım" %1-2).

Yazılım vs Donanım

- Yazılım eskimez.
- Oysa, her donanımın belli bir ömrü vardır. Ömrünü tamamlayan donanım yenisi ile değiştirilir.
- Yazılımın eskimesi ortaya çıkabilecek yeni ihtiyaçları karşılayamaması, kullandığı teknolojinin eskimesi olarak tanımlanabilir.
- Yeni gereksinimler yazılıma ekler yaparak yansıtılır.

Yazılım vs Donanım

- Yazılım en az donanım kadar önemlidir.
  - Diyaliz makinelerinde kullanılan yazılımların 2000 yılı uyumsuzluğundan ötürü, bir çok diyaliz makinesi çalışamamış ve böbrek hastaları zor durumda kalmıştır.
  - Japonya'da telefon yazılımında ortaya çıkan bir yazılım hatası onbinlerce abonenin saatlerce telefon konuşması yapamamasına neden olmuştur.

Yazılım ve Donanım

- Yazılım kopyalama ve donanım kopyalama farklıdır.
  - Hata toleransı amacıyla, hayati olan bir donanımın sistemde bir kopyası daha bulundurulur ve sistemde biri arızalandığında diğeri çalışmayı devralabilir.
  - Oysa, bir yazılımı sistemde iki ayrı bilgisayar üzerine kopyalamak oluşabilecek hatalara çözüm olmayacaktır. Belki, sisteme aynı işi yapan iki farklı eş yazılım yüklenmesi çözüm olabilir (kritik yazılım sistemleri-uçak avionics).

Tipik Bir Yazılım Üretim Ortamı

- Değişik yetenekte bir çok personel (analist, programcı, test uzmanı, vs.)
- Yazılım çıktısı ile ilgilenen kullanıcılar
- Yeniliğe tepki gösteren kullanıcılar ve yöneticiler !
- Yeterince tanımlanmamış kullanıcı beklentileri
- Personel değişim oranının yüksekliği
- Yüksek eğitim maliyetleri
- Dışsal ve içsel kısıtlar (zaman, maliyet, işgücü, vs)
- Standart ve yöntem eksiklikleri
- Verimsiz kaynak kullanımı
- Mevcut yazılımlardaki kalitesizlik
- Yüksek üretim maliyeti

Yazılım Mühendisliği

- IEEE Tanımı (1993)

"Yazılım Mühendisliği:

Sistemli, düzenli, ölçülebilir bir yaklaşımın

yazılım geliştirmede,

yazılımın işlenilmesinde ve

bakımında uygulanmasıdır.

- *Diğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.*

Yazılım Mühendisliği

- Yazılım üretiminin mühendislik yöntemleriyle yapılmasını öngören ve bu yönde;
  - yöntem,
  - araç
  - teknik ve
  - metodolojiler

  üreten bir disiplindir.

Yazılım Mühendisliği

- Yazılım mühendisliği bir yöntemler, teknikler ve araçlar kümesi olarak değerlendirilebilir.
- Yazılım mühendisliğinin hedefi; yazılım üretimindeki karmaşıklıkları gidermektir.
- Geçmişte kullanılan iş akış şemaları gibi yöntemler günümüzde yetersiz kalmaktadır.
- Ayrıca, yazılım üretimi işi tek kişinin başarabileceği boyuttan çıkmış ve bir takım işi biçimine dönüşmüştür.

Yazılım Mühendisi

- Yazılım Mühendisliği İşini yapan kişidir.
- Temel hedefi; üretimin en az maliyet ve en yüksek nitelikte yapılmasını sağlamaktır.
- Programcı değildir. Ancak programcının tüm yeteneklerine sahiptir.
- Yazılımın daha çok mantıksal boyutuyla ilgilenir ve işi insanlarla ilişkiyi gerektirir.
- Sistem analisti de değildir. Farkı; analist sadece sistemin analiz aşaması ile ilgilenirken, yazılım mühendisi tüm aşamaların içindedir.

Yazılım Hataları

- Bir programı tüm ayrıntıları ile test etmek teorik olarak mümkün olmakla birlikte, uygulamada bu mümkün değildir.
- Yazılım ancak sınırlı sayıda veri ile sınanabilir.

Mantıksal Tasarım	%20
İşlevsel Tasarım	%15
Kodlama	%30
Belgeleme ve Diğerleri	%35

Yazılımda Hata Düzeltme Maliyetleri

- Yazılım üretimindeki hatalar yayılma özelliği gösterir.
- Bu nedenle, hata düzeltme maliyetleri ilerleyen aşamalarda giderek artar.

Analiz	1
Tasarım	5
Kodlama	10
Test	25
Kabul Testi	50
İşletim	100

Yazılım Maliyetleri

- Yazılım = $ 100.000
- Donanım = $ 1000

Yazılım Sistemlerinin Sınıflandırılması

- İşlevlerine göre
- Zamana dayalı özelliklere göre
- Boyuta göre

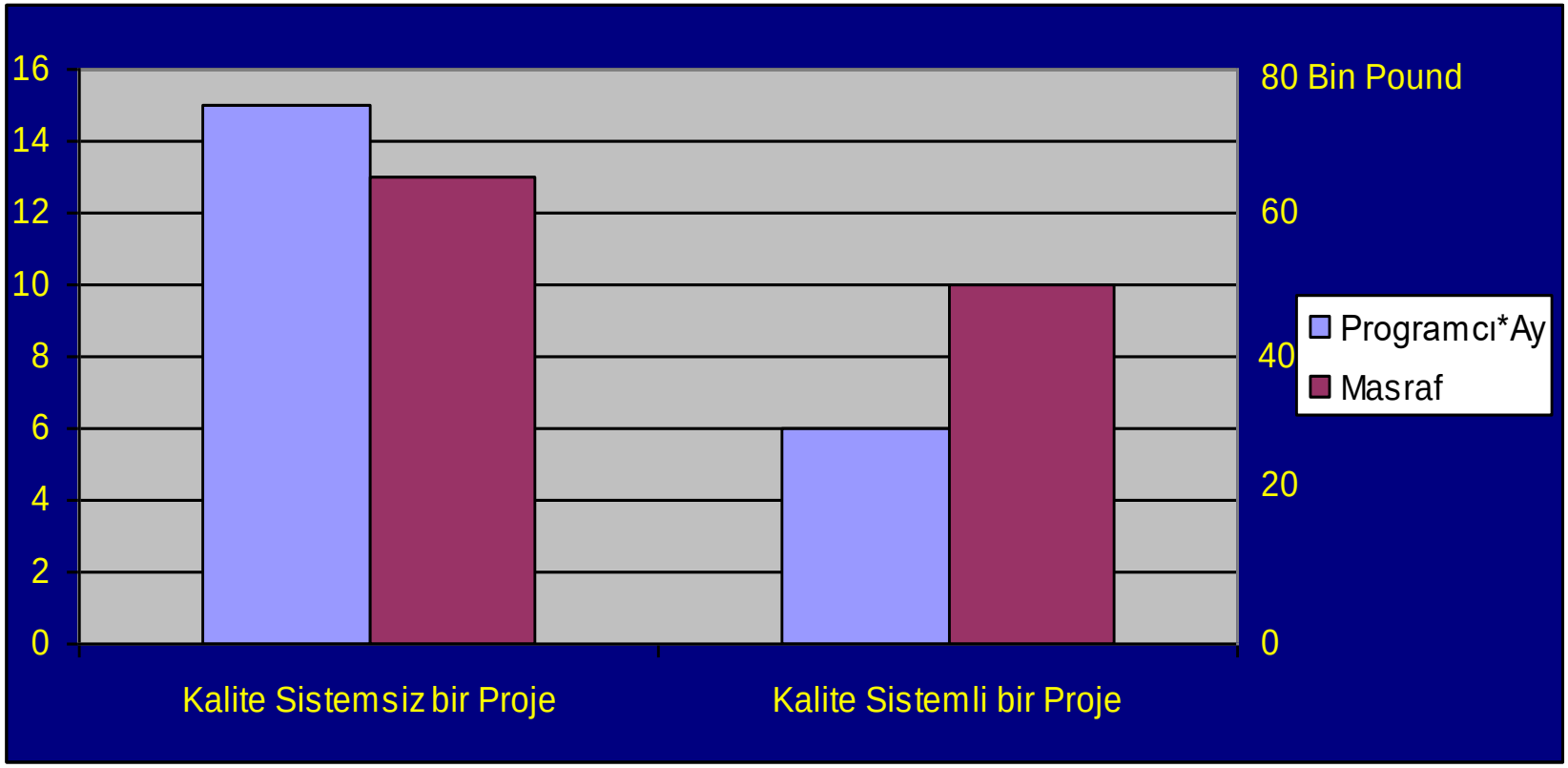
İşleve Göre Sınıflandırma

Hesaplama	Mühendislik Çözümleme
Veri İşleme	Bankacılık
Süreç Temelli	Gömülü Sistemler
Kural Temelli	Robotik, Yapay Zeka
CAD	Sinyal İşleme

Boyuta Göre Sınıflandırma

Küçük (SS<2000)	PC Oyunları Öğrenci Projeleri
Orta (2000<SS<100,000)	CAD BDE Yazılımları
Büyük(100,000<SS<1 Milyon)	İşletim Sistemleri
Çok Büyük (SS>1 Milyon)	Komuta Kontrol Sistemleri Hava Tahmini Sistemleri Yıldız Savaşları Sistemleri

Yazılımda Kalite



Yazılımda Kalite

- Üretim Süreci Boyunca ara ürünlere ilişkin kalite standartlarının geliştirilmesi ve geliştirme işlemlerinin bu standartlara uygunluğunun denetlenmesidir.
- Yazılım kalite sağlama etkinlikleriyle;
  - Yazılım maliyetleri düşürülür,
  - Yazılım üretiminin yönetimi kolaylaşır,
  - Belgeleme ve standart sorunları giderilir.

Yazılımda Kalite

Ekonomi	Tamlık	Yeniden Kullanılabilirlik	Etkinlik	Bütünlük
Güvenirlik	Modülerlik	Belgeleme	Kullanılabilirlik	Temizlik
Değiştirilebilirlik	Geçerlik	Esneklik	Genellik	Sınanabilirlik
Taşınabilirlik	Bakılabilirlik	Anlaşılabilirlik	Birlikte Çalışabilirlik	

Temel terminoloji

- Yazılım (**Software**):
- Yazılım sadece bir bilgisayar programı değildir.
- Basılı veya elektronik ortamdaki her tür dokümanı da içeren ürün.
- Dokümanlar yazılım mühendislerine ve son kullanıcıya yönelik olabilir.
- Uygulama (**Application**) kelimesi de kullanılabilir
- Yazılım ve Donanım adlandırması:
- Yazılım: Software (SW)
- Donanım: **Hardware** (HW)
- İngilizce adlandırma, yazılımın kolaylıkla değiştirilebilecek, oyun hamuru gibi yumuşak bir şeyler olduğu kanısına yöneltir.
- Ancak yazılım daha çok kil veya cam gibidir, bir kere tamamlandıktan sonra değiştirmesi zordur.

Yazılım Türleri

Sistem Yazılımı: System Software

- Diğer programlara hizmet sunmak üzere hazırlanmış programlar.
- Derleyiciler, işletim sistemleri, vb.
- Karmaşık olsa bile belirli, iyi tanımlanmış bilgi yapıları ile uğraşır.
- **Mühendislik Yazılımı / Bilimsel Yazılım : Engineering / Scientific Software**
- Mühendislik ve bilimsel hesaplamalarda kullanılmak üzere hazırlanmış programlar.
- "Numara öğütmek / Number crunching": Bu tip programlar büyük hacimli verilerle uğraştığından bu deyimle karşılaşabilirsiniz.

Yazılım Türleri

Şirket Yazılımı / Kurumsal Uygulamalar (Enterprise software):

- Belirli ticari iş gereksinimlerine yönelik programlar.
- İş süreçleri (business process) ile ilgili bilgiye sahip olmalıdır.
- Genellikle müşteriye özel tasarlanır.
- Veri dönüştürme ve değerlendirme uygulamaları, iş süreçlerinin kimi zaman gerçek zamanlı izlenilmesi, vb.
- Uygulama Yazılımı (Application software):
- Product-line, shrink-wrapped, off-the-shelf, vb.
- Farklı müşteriler tarafından kullanılabilecek genel amaçlı yazılımlar
- Cari hesap uygulamaları, çeşitli otomasyon programları, kelime işlem uygulamaları, vb.

Yazılım Türleri

Gömülü (Embedded) Yazılım :

- Bir ürün veya sistemin bir parçası olup, bu sistemin kendisi ve/veya son kullanıcısı için denetim işlemleri yürüten programlar.
- Gerçek zamanlı (Real Time) uygulamalardır.
- Programın yanıt verme / tepki süresinin (response time) belli bir zaman aşımını (timeout) geçmemesi gerekir.
- Öyle ki, çok hassas bir yanıtın geç gelmesi yerine, yeterli bir yanıtın çabuk gelmesi daha önemli olabilir.

Yazılım Türleri

Ağ Uygulamaları (Web applications):

- Ağ üzerinden haberleşerek hizmet almaya veya vermeye yönelik uygulamalar.
- Şirket yazılımları ile etkileşimde bulunabilirler.
- E-ticaret, B2B, B2C, web servisleri, web tarayıcıları, vb.
- B2B: Business to Business
- İki veya daha fazla ticari firma arasında çalışan uygulamalar.
- Birden çoğa, çoktan çoğa, işbirliğine ve ticari işlemlere yönelik
çeşitli uygulamalardır.
- B2C: Business to Customer
- Doğrudan son kullanıcıya satış amaçlı.

Yazılım Türleri

Yapay Zeka (Artificial Intelligence: AI) Yazılımları :

- Sayısal olmayan algoritmalarla karmaşık sorunları çözmeye yönelik yazılımlar.

- Robotik, uzman sistemler (expert systems), örüntü tanıma (pattern recognition) (ses ve görüntü), vb.

Yazılım Türleri

Eski Yazılım (Legacy Software)

- İş sürecinin önemli bir parçası olan ve çok uzun süredir kullanılan yazılımlar.
- Şirketler, yazılım sistemleri dahil, yaptıkları yatırımı mümkün olan en uzun sürece kullanmak ister.
- Ancak iş alanındaki gereksinimler hızla değişebilir.
- Yazılım artık yeni ihtiyaçları karşılayacak şekilde esnetilemiyorsa, yazılım yeniden tasarlanmalıdır.
- Eski uygulamaya şirketin diğer bir çok süreci ve bilgi sistemi bağımlı ise, tümleştirme (integration) çalışmaları zor olabilir.

Yazılım Yaşam Döngüsü

Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen aşamalardır.

- Döngü: Kullanım dışı bırakılan yazılımın yerine yenisi hazırlanabilir.
- Döngünün aşamalarının belirlenmesi ve tanımlanması ile yazılım geliştirme modelleri/süreçleri elde edilir.

Yazılım Geliştirme Süreçleri / Modelleri

Yazılım geliştirme bir süreçtir (**sw development** process)

- Süreç: Önceden belirlenmiş adımlardan oluşan iş akışı.
- Yazılım geliştirme modelleri, sürecin yapısını ve adımlarını belirler.

Yazılım Geliştirme Süreçleri

Modellerin tanımladığı adımlar arasında farklar olmakla beraber, her süreç modelinde bulunan genel işlemler şu şekildedir:

- Çözümleme (Analysis)
- Ne yapılacak?
- Tasarım (Design)
- Nasıl yapılacak?
- Gerçekleme (Implementation)
- Haydi yap!
- Sınama (Testing)
- Doğru yaptın mı?
- Bakım (Maintenance)
- Değişmeyen tek şey değişimin kendisidir!

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği

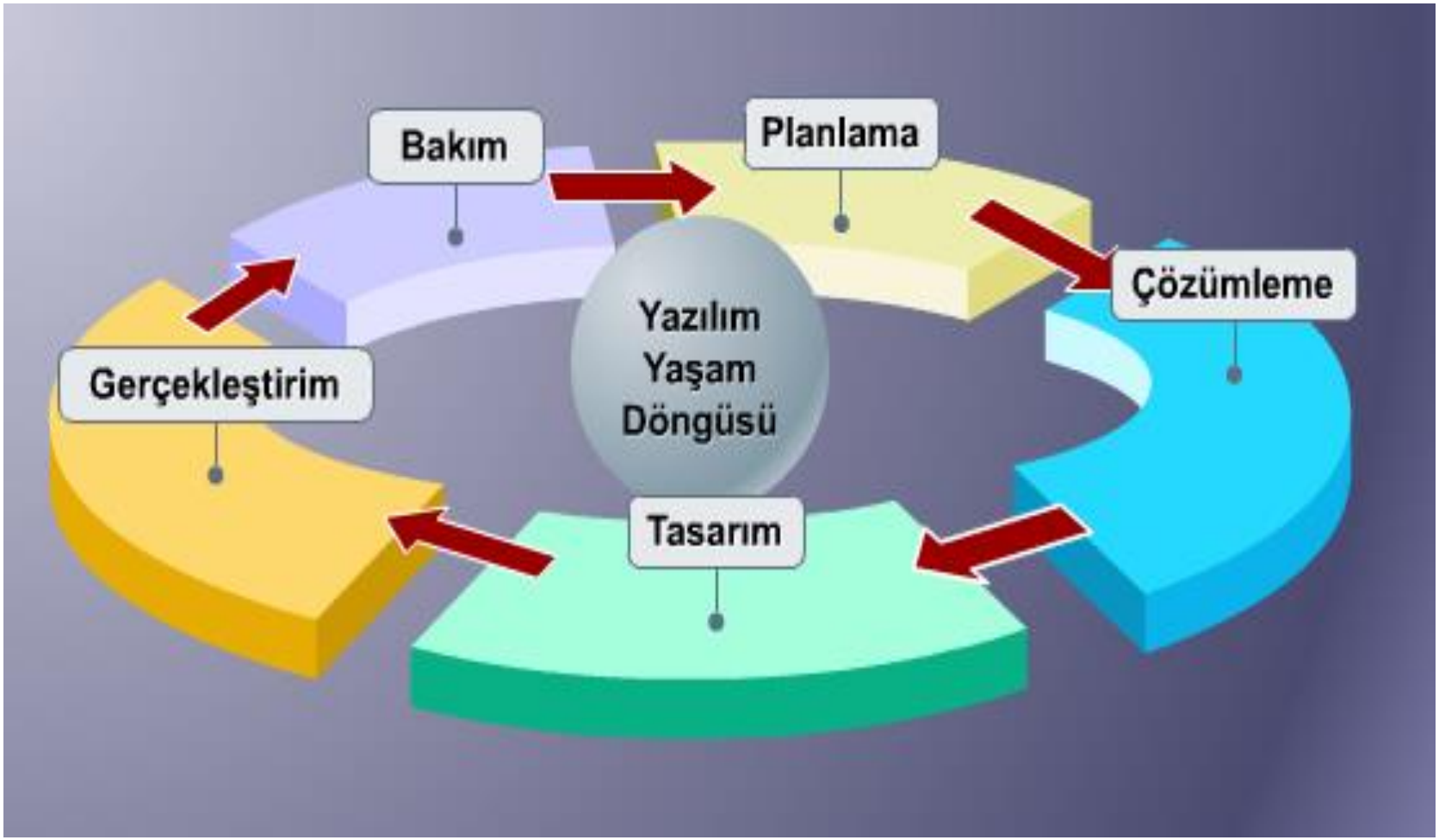
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ-Hafta 2

Hedef

- Bu bölümde, yazılım geliştirme yaşam döngüsünün çekirdek aşamaları olan, **planlama**, **çözümleme**, **tasarım**, **gerçekleştirim** ve **bakım** aşamaları tanıtılmakta, yaşam döngüsü modelleri ya da süreç modelleri olarak da **yaşam döngüsü**nün (life cycle) uygulamada kullanım modellerinin açıklanması hedeflenmektedir.
- Metododoloji, kavramı bu bölümde açıklanmaktadır.

Yazılım Yaşam Döngüsü

- Planlama, çözümleme, tasarım, gerçekleştirim, bakım

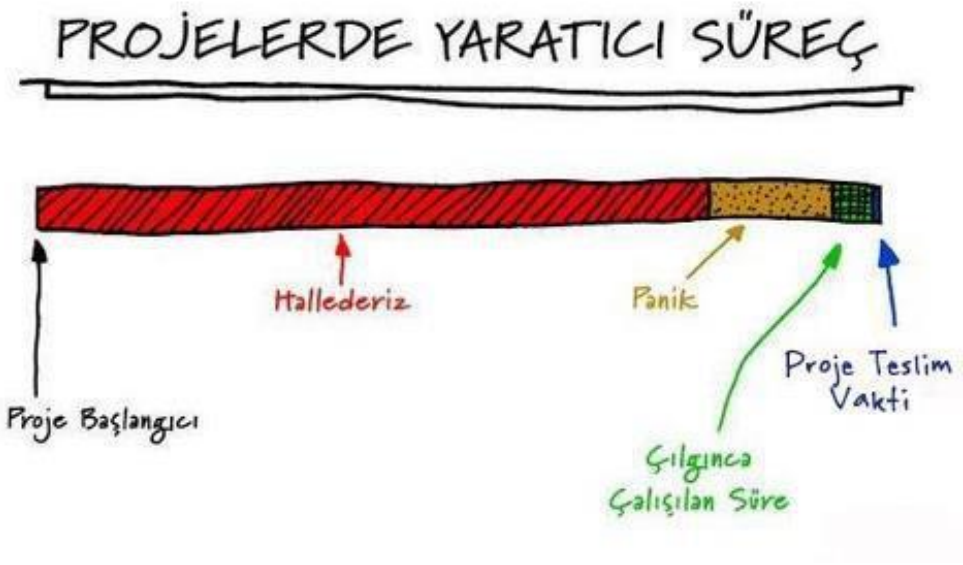


Yazılım Yaşam Döngüsü

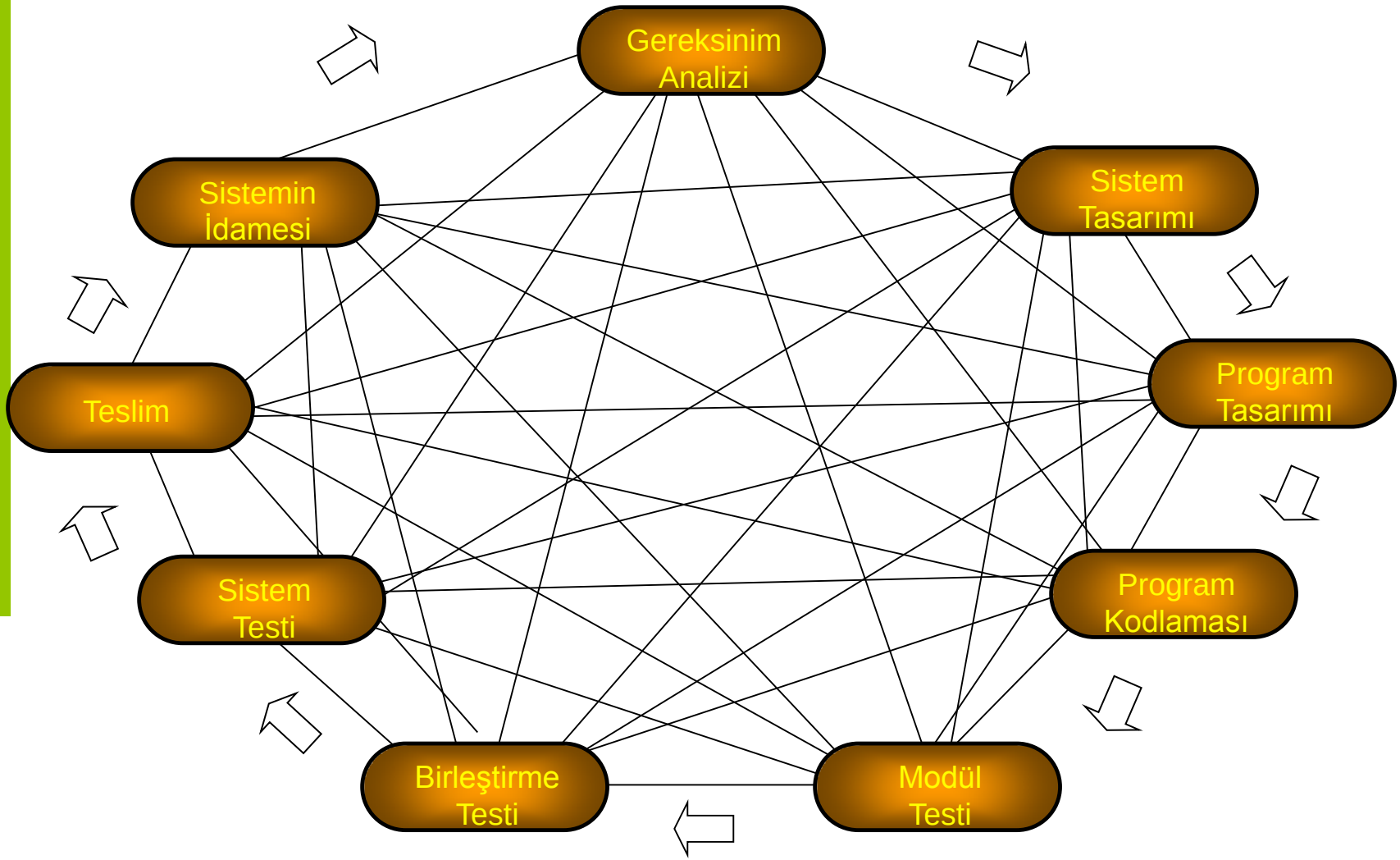
- Herhangi bir yazılımın, **üretim aşaması** ve **kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
- Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
- Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.
- Bu nedenle yazılım yaşam döngüsünün tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü-Ne sağlar?

- Mühendislik faaliyetlerinin (üretim, işletme ve bakım aşamaları) yazılıma uyarlanması
- Üretimden kullanıma kadar tüm sürecin fazlara ayrılması ve böylece yönetim kolaylığı
- Her bir aşamada ne yapılacak?
- Her bir aşamadaki standartlara uyumluluk ile kaliteli yazılım geliştirme vb.



Gerçek Hayatta Program Geliştirme



Yazılım Yaşam Döngüsü Temel Adımları

Planlama: Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite (olurluk-yapılabilirlik) çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

Analiz: Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Mevcut var olan işler incelenir, temel sorunlar ortaya çıkarılır.

Tasarım: Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.

mantıksal; önerilen sistemin yapısı anlatılır,olası örgütsel değişiklikler önerilir.

fiziksel; yazılımı içeren bileşenler ve bunların ayrıntıları.

Gerçekleştirim: Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.

Bakım: Hata giderme ve yeni eklentiler yapma aşaması (teslimden sonra).

Yazılım Yaşam Döngüsü Temel Adımları

- Yaşam döngüsünün temel adımları çekirdek süreçler (core processes) olarak da adlandırılır.

- Bu süreçlerin gerçekleştirilmesi amacıyla
  - belirtim (specification) yöntemleri - bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler
  - Süreç (process) modelleri - yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlayan modeller

kullanılır.

Belirtim Yöntemleri

- Süreç Akışı İçin Kullanılan Belirtim Yöntemleri

  Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler (Veri Akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları).

- Süreç Tanımlama Yöntemleri

  Süreçlerin iç işleyişini göstermek için kullanılan yöntemler (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).

- Veri Tanımlama Yöntemleri

  Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemler (Nesne İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü).

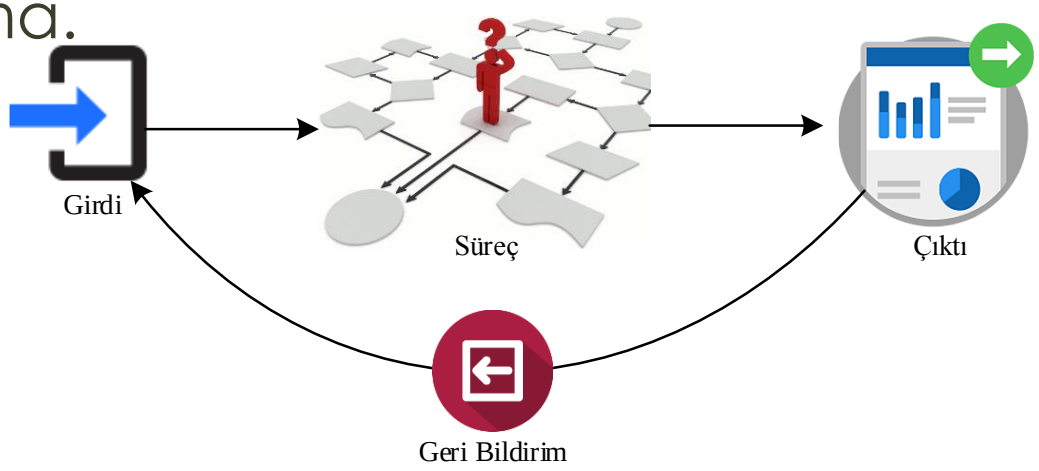
Yazılım Süreci Modelleri

- Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir.
- Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.

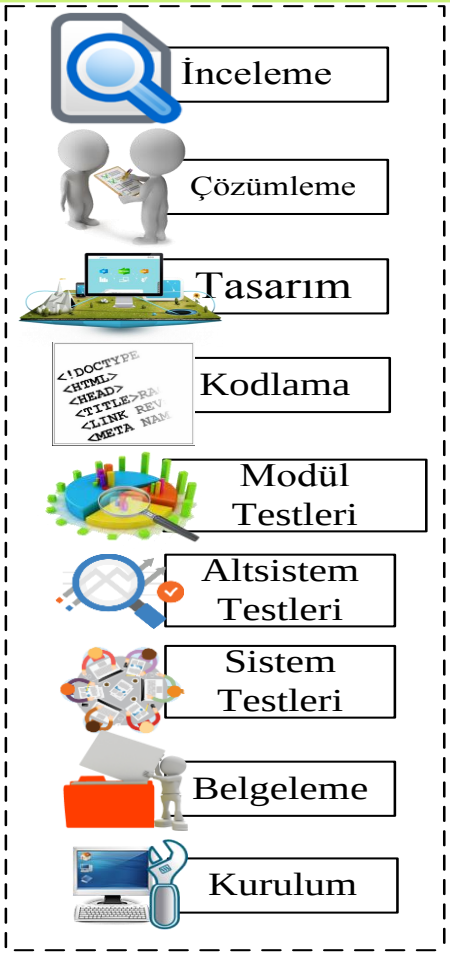
1. Gelişigüzel Model
2. Barok Modeli
3. Çağlayan (Şelale) Modeli
4. V Modeli
5. Helezonik (Spiral) Model
6. Evrimsel Model
7. Artırımsal Model
8. Araştırma Tabanlı Model

Gelişigüzel Model

- Herhangi bir model ya da yöntem yok.
- Geliştiren kişiye bağımlı (belli bir süre sonra o kişi bile sistemi anlayamaz hale gelebilir ve geliştirme güçlüğü yaşar).
- İzlenebilirliği ve bakımı oldukça zor.
- 60'lı yıllarda kullanılan ilkel model
- Genellikle tek kişilik üretim ortamı.
- Basit programlama.



Barok Modeli



Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.

70'li yıllar.

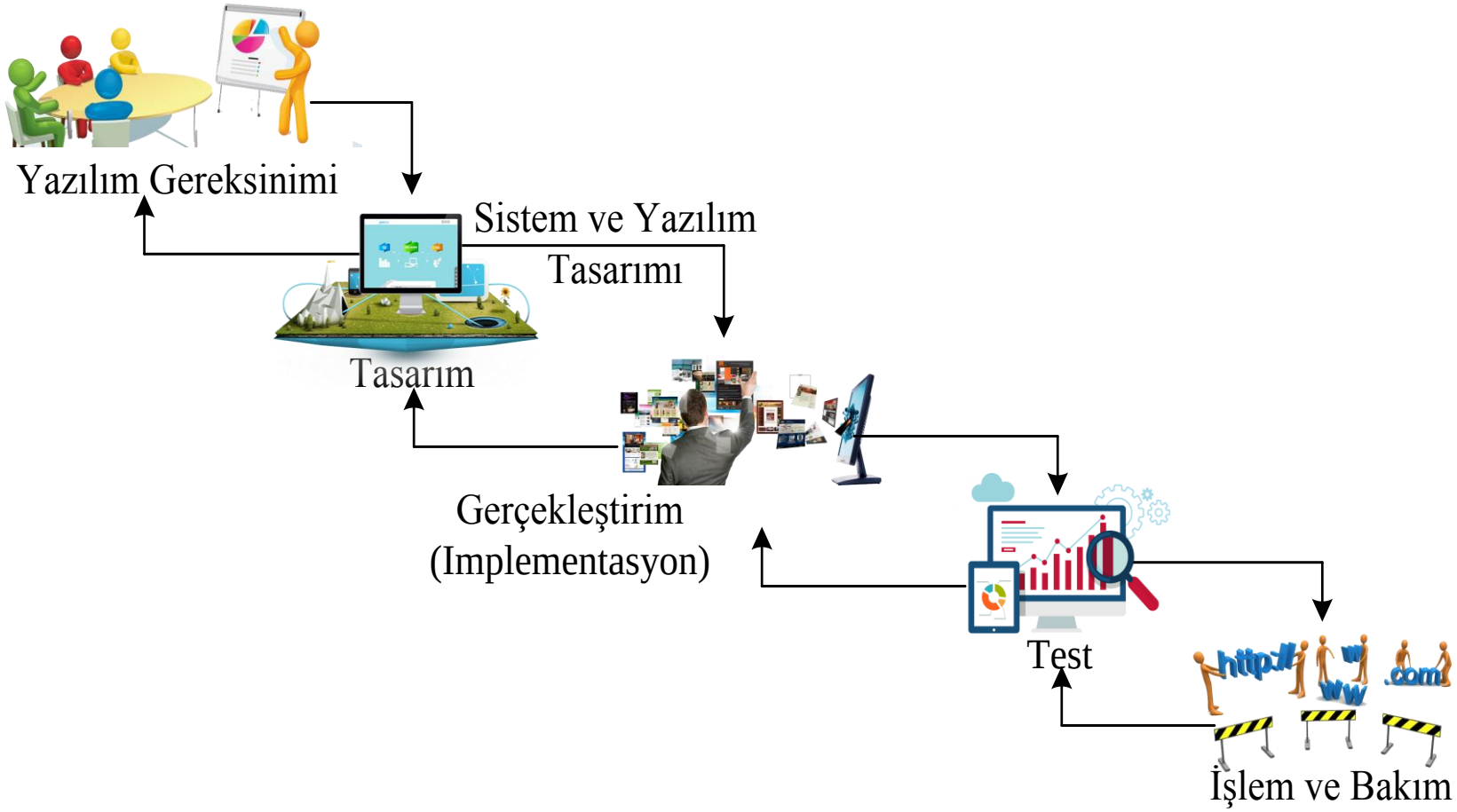
Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasını öngörür.

Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.

Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.

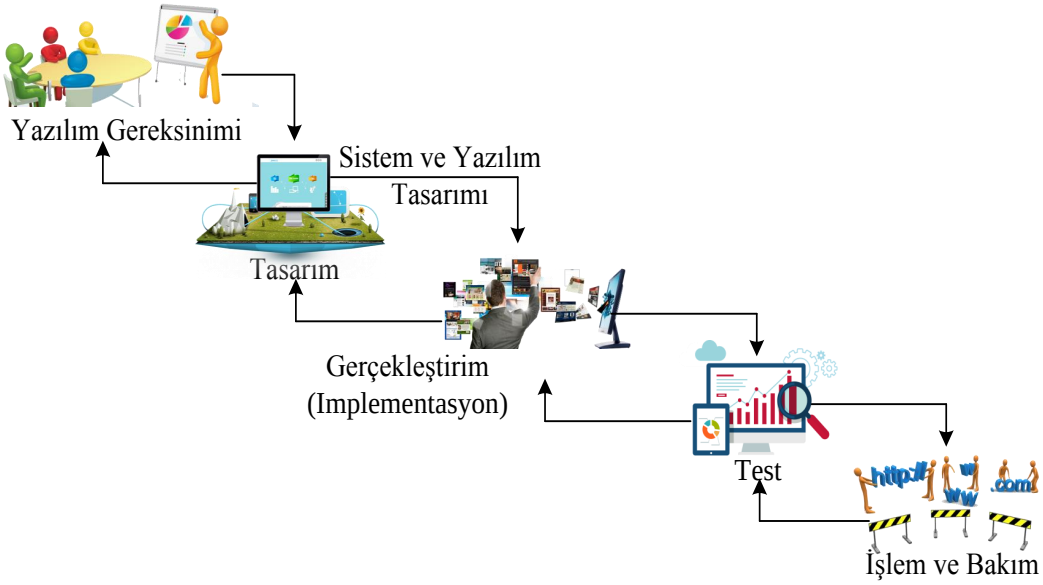
Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir

Çağlayan (Şelale) Modeli



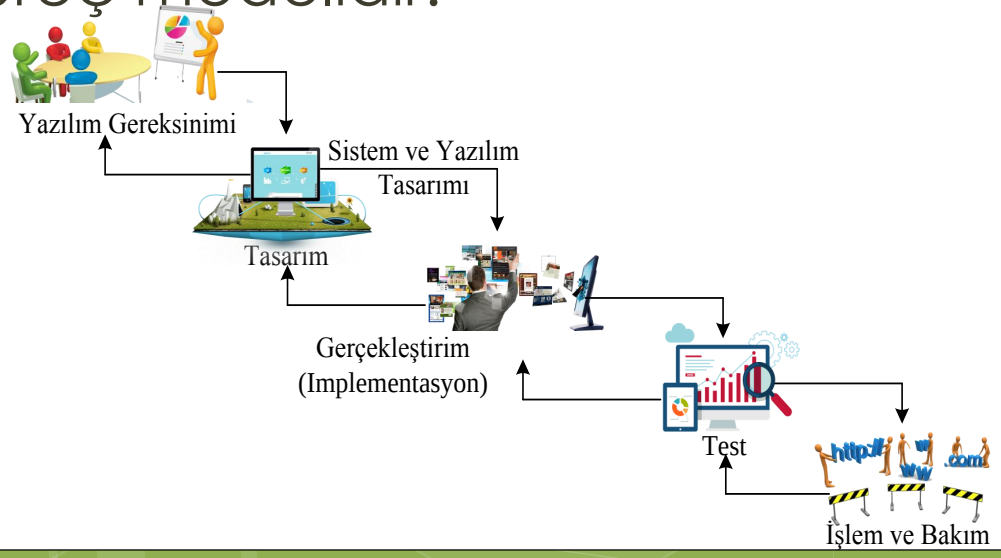
Çağlayan (Şelale) Modeli

- Yaşam döngüsü temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir.
- İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
- Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.



Çağlayan (Şelale) Modeli

- Barok modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.
- Barok modele göre geri dönüşler iyi tanımlanmıştır.
- Yazılım tanımlamada belirsizlik yok (ya da az) ise ve yazılım üretimi çok zaman almayacak ise uygun bir süreç modelidir.



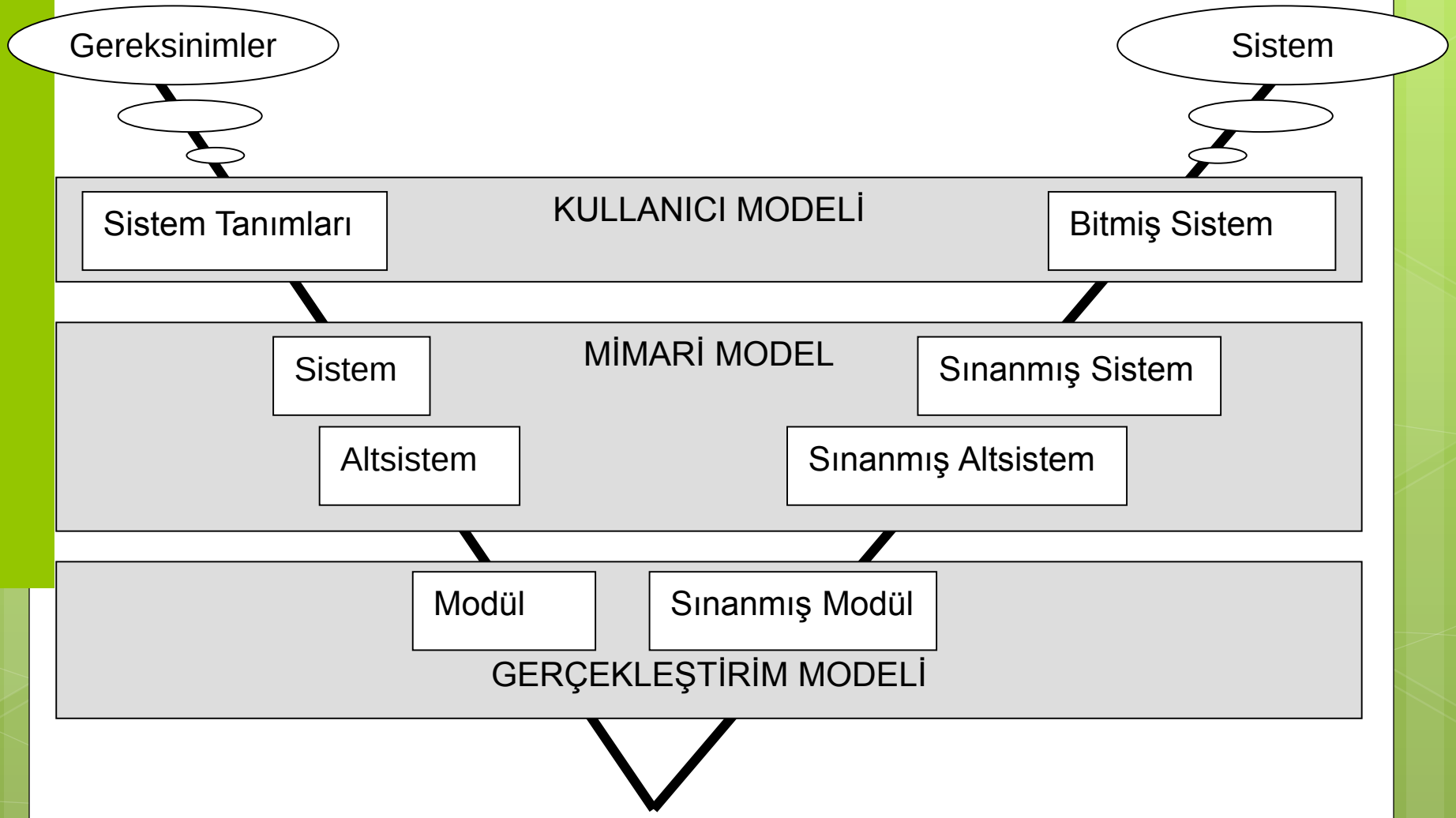
Avantajları

- Müşteriler ve son kullanıcılar tarafından da iyi bilinen anlaşılabilen adımlardan oluşur.
- İterasyonlar (tekrarlamalar) bir sonraki ve bir önceki adımlarla gerçekleşir, daha uzak adımlarla olması nadirdir.
- Değişiklik süreci yönetilebilir birimlere bölünmüştür.
- Gereksinim adımı tamamlandıktan sonra sağlam bir temel oluşur. Aşamaları iyi anlaşılabilir.
- Proje yöneticileri için işin dağılımını yapma açısından kolaydır.
- Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
- Kalite gereksinimlerinin bütçe ve zaman kısıtlamasına göre çok daha önemli olduğu projelerde iyi çalışır.

Sorunları

- Gerçek yaşamdaki projeler genelde yineleme gerektirir.
- Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.
- Gereksinim tanımlamaları çoğu kez net bir şekilde yapılamadığından dolayı, yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.
- Yazılım üretim ekipleri bir an önce program yazma, çalıştırma ve sonucu görme eğiliminde olduklarından, bu model ile yapılan üretimlerde ekip mutsuzlaşmakta ve kod yazma dışında kalan (ve iş yükünün %80'ini içeren) kesime önem vermemektedirler.
- Üst düzey yönetimlerin ürünü görme süresinin uzun oluşu, projenin bitmeyeceği ve sürekli gider merkezi haline geldiği düşüncesini yaygınlaştırmaktadır.

V Süreç Modeli



V Süreç Modeli

Sol taraf üretim, sağ taraf sınama işlemleridir.

V süreç modelinin temel çıktıları;

- Kullanıcı Modeli

  Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

- Mimari Model

  Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

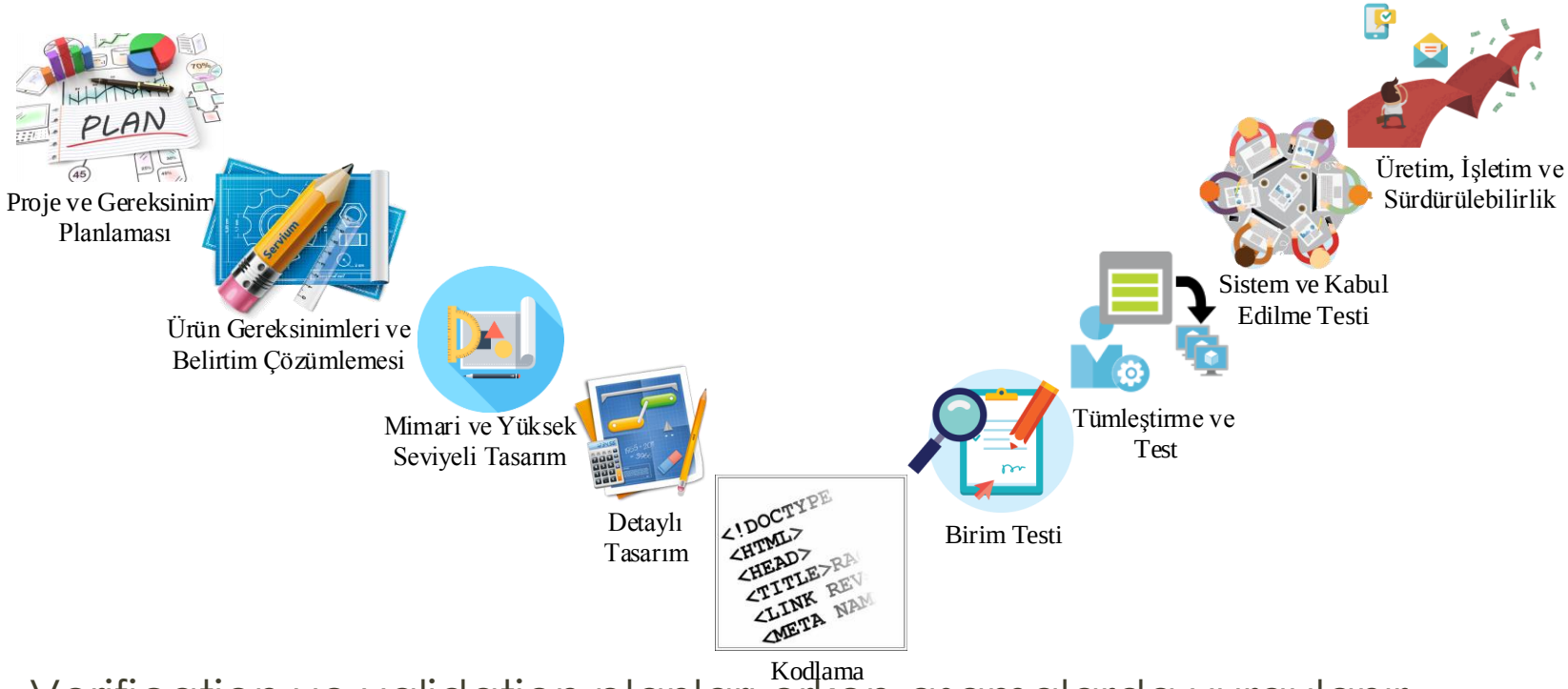
- Gerçekleştirim Modeli

  Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

V Süreç Modeli

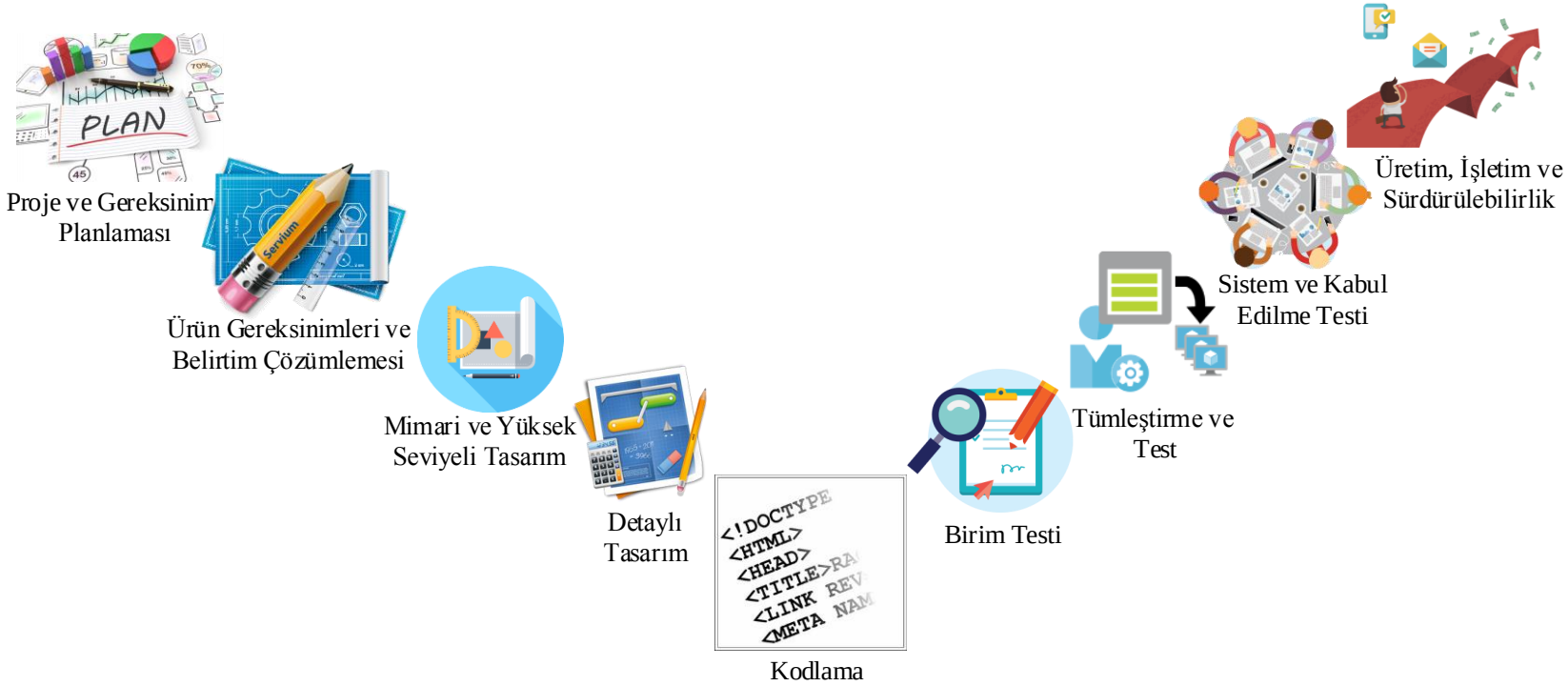
- Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygun bir modeldir.
- Model, kullanıcının projeye katkısını arttırmaktadır.
- BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur:
  - İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta,
  - İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçeklenmektedir.

V Modeli-Avantajlar



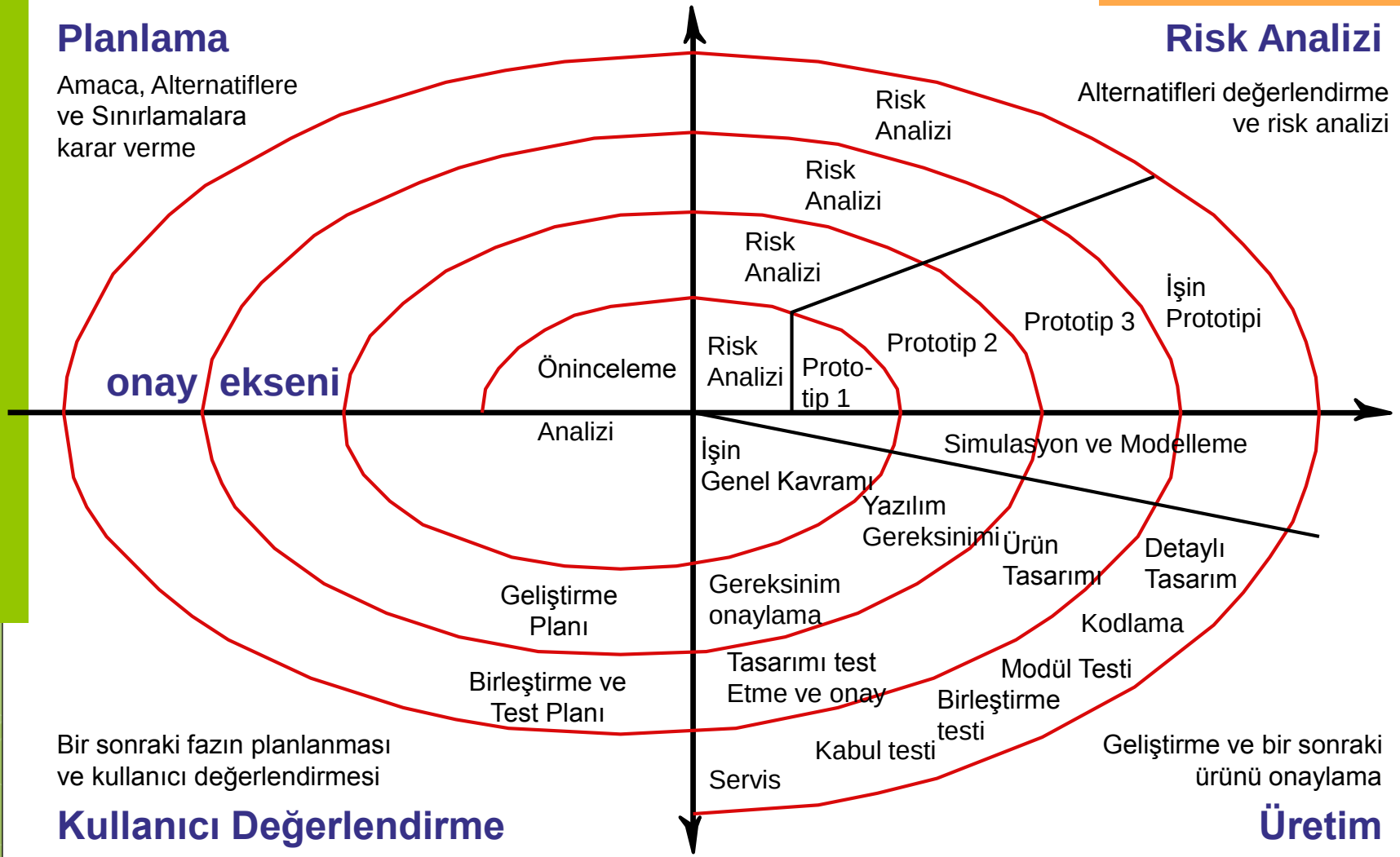
- Verification ve validation planları erken aşamalarda vurgulanır.
- Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
- Proje yönetimi tarafında takibi kolaydır
- Kullanımı kolaydır.

V Modeli-Dezavantajlar



- Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz.
- Aşamalar arasında tekrarlamaları kullanmaz.
- Risk çözümleme ile ilgili aktiviteleri içermez.

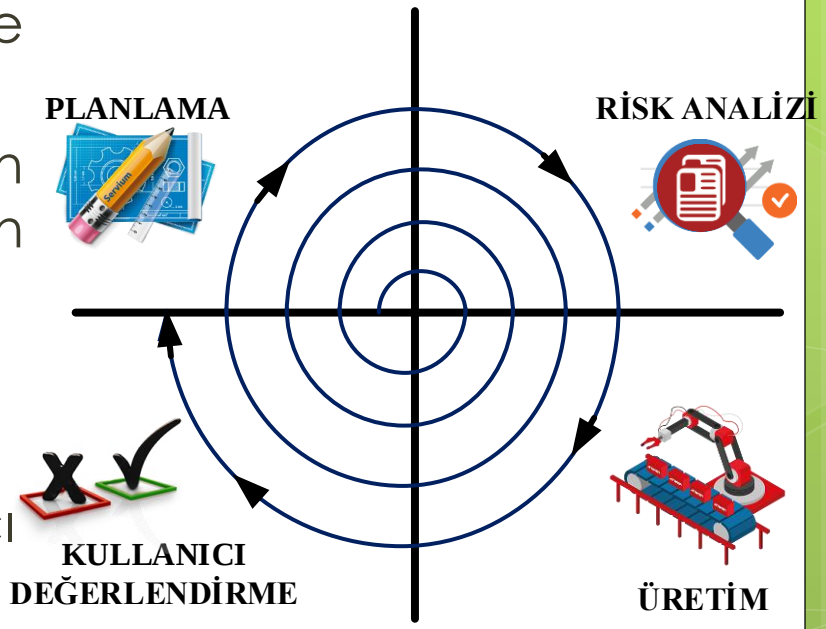
Helezonik(Spiral) Modeli



Helezonik Model

1. **Planlama:** Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme
2. **Risk Analizi:** Risk seçeneklerinin araştırılması ve risklerin belirlenmesi
3. **Üretim:** Ara ürünün üretilmesi
4. **Kullanıcı Değerlendirmesi**

   Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler



Helezonik 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.

Helezonik Model

- 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.

Prototipleme

- Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.
- Gereksinimler netleştikçe prototip kullanıcı ihtiyaçlarına göre düzeltilir
- Müşteri memnun olana kadar düzeltmelere devam et

Prototipleme-Avantajları

- Kullanıcı sistem gereksinimlerini görebilir
- Karmaşa ve yanlış anlaşılmaları engeller
- Yeni ve beklenmeyen gereksinimler netleştirilebilir
- Risk kontrolü sağlanır

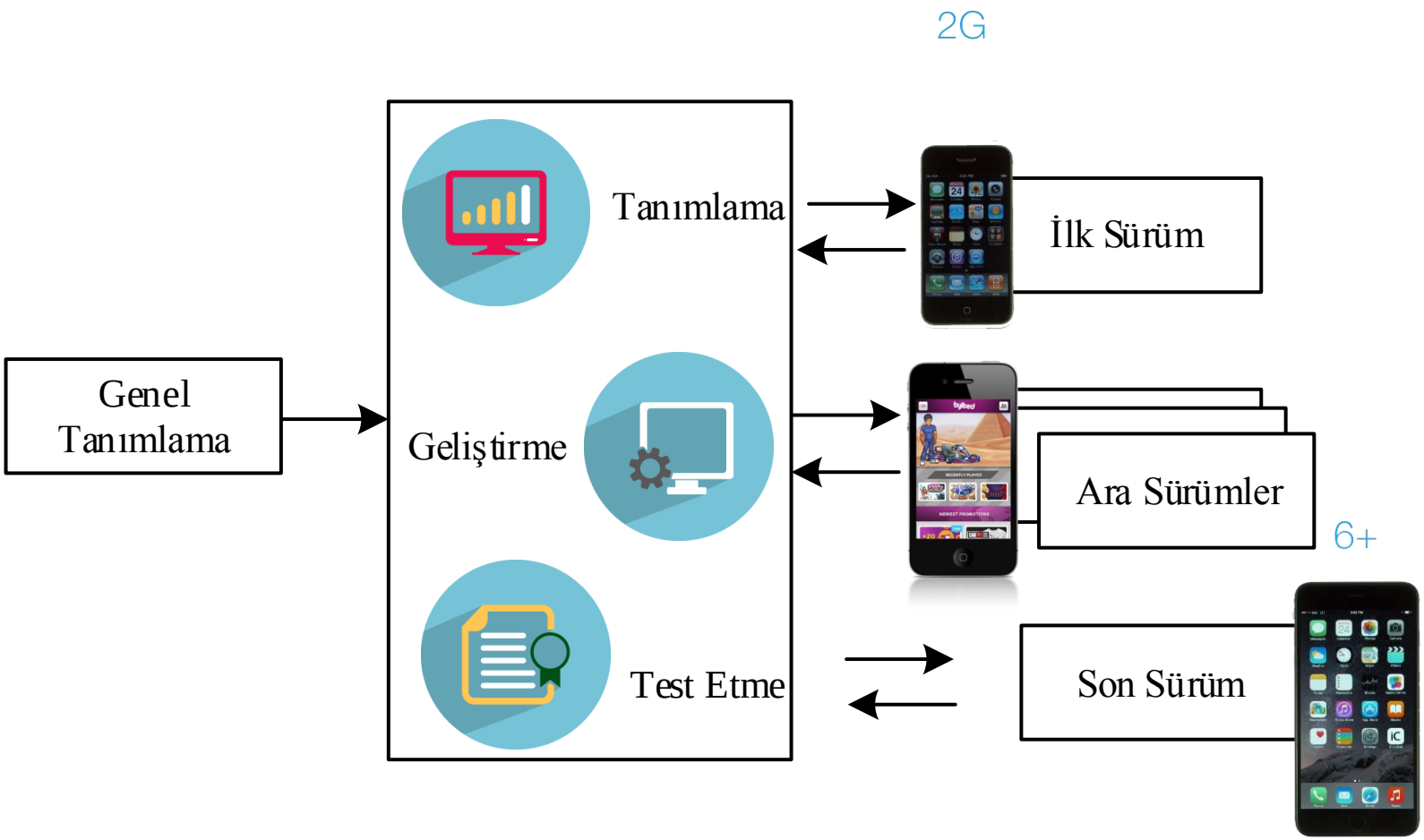
Prototipleme-Dezavantajları

- Belgelendirmesi olmayan hızlı ve kirli (quick and dirty) prototipler
- Prototip hedefleri net değilse kod hackleme ya da jenga başlar
- Düzeltme aşaması atlanırsa, düşük performansa yol açar
- Müşteri prototipten de son ürün gibi görünüm ve etki bekler

Evrimsel Geliştirme Süreç Modeli

- 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.

Evrimsel Geliştirme Süreç Modeli



Örnek

- Çok birimli banka uygulamaları.
- Önce sistem geliştirilir ve Şube-1'e yüklenir.
- Daha sonra aksaklıklar giderilerek geliştirilen sistem Şube-2'ye yüklenir.
- Daha sonra geliştirilen sistem Şube-3'e,…. yüklenir.
- Belirli aralıklarla eski şubelerdeki güncellemeler yapılır.

Evrimsel Geliştirme Süreç Modeli

İki çeşit evrimsel geliştirme vardır:

Keşifçi geliştirme (exploratory development)

- Hedef: Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmek
- İyi anlaşılan gereksinimlerle başlanmalıdır.

“ Ne istediğimi sana söyleyemem ama onu gördüğümde bilirim”

Atılacak prototipleme (throw-away prototyping)

- Hedef: Sistem gereksinimlerini anlamak
- Tam anlaşılmamış gereksinimlerle başlar

Aksaklığı

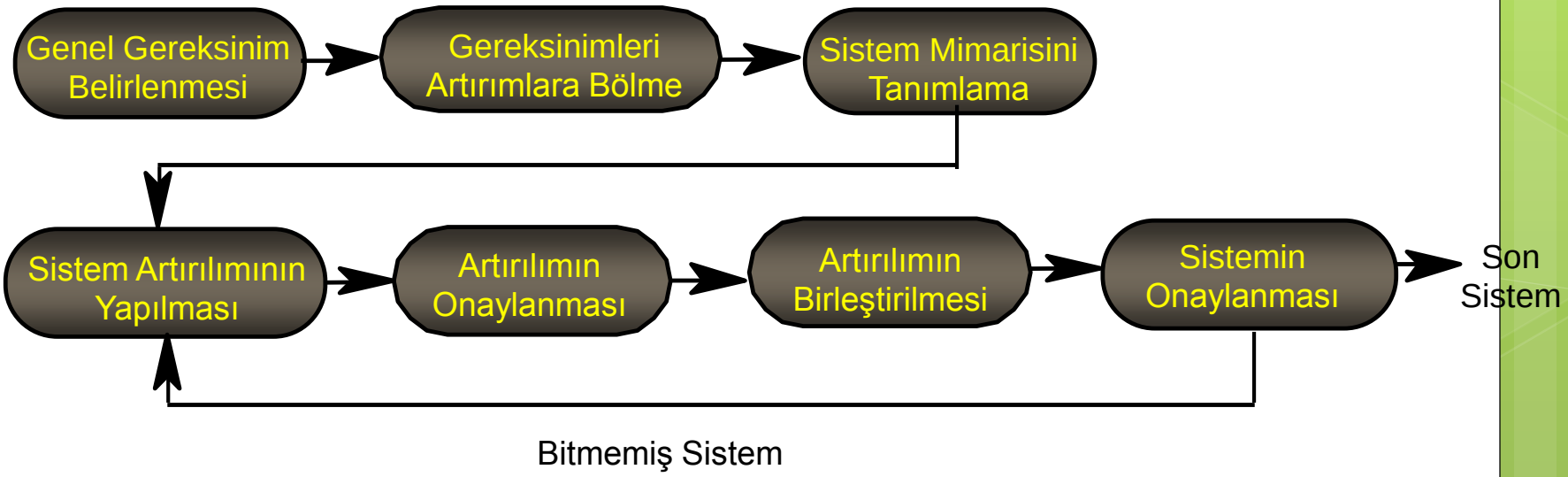
- Değişiklik denetimi
- Konfigürasyon Yönetimidir
  - Sürüm Yönetimi
  - Değişiklik Yönetimi
  - Kalite Yönetimi

Artırımsal Geliştirme Süreç Modeli

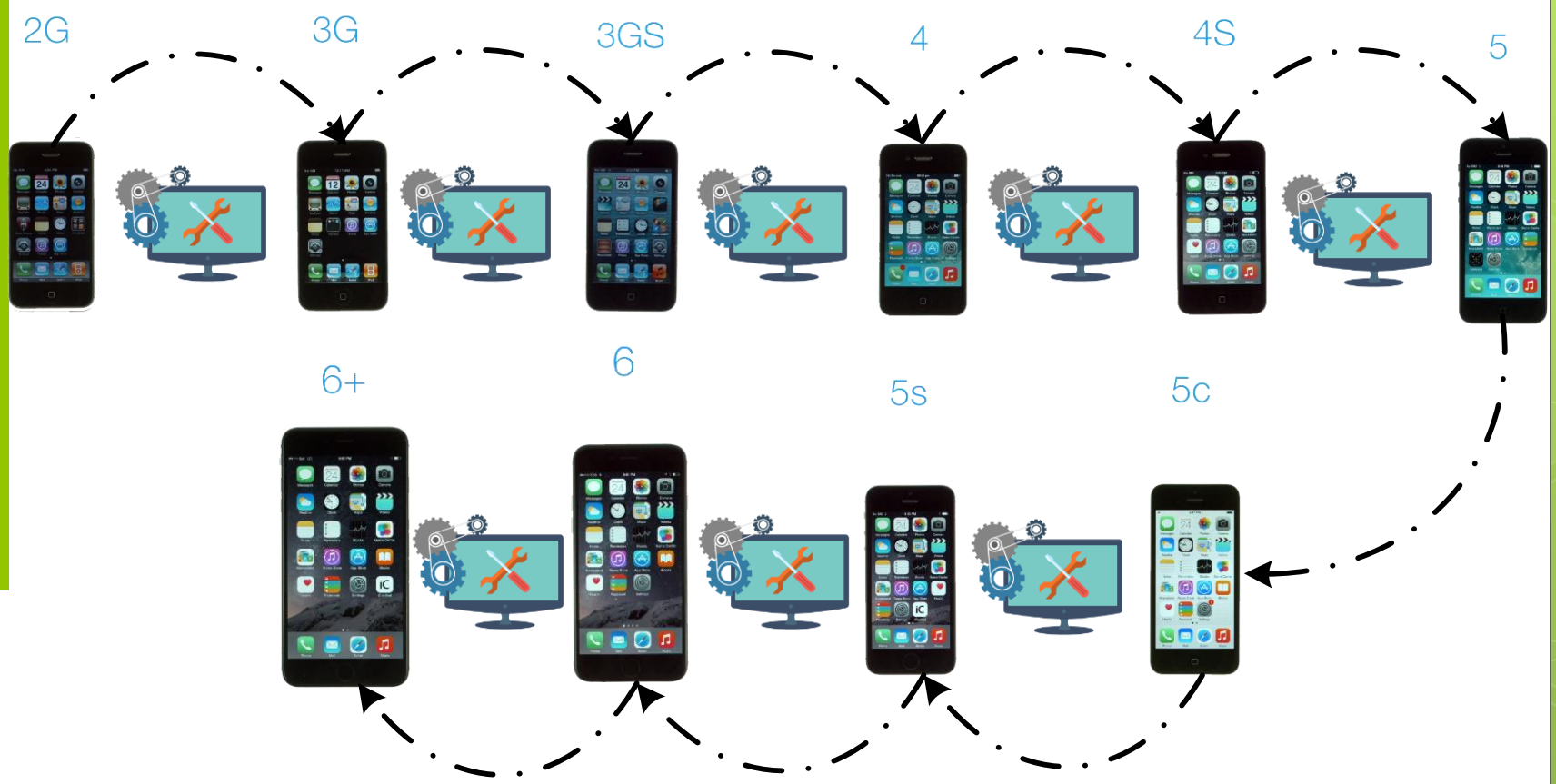
- Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.
- Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 2 haftada bir gelişiminin izlenmesi (bitirme tezleri).
- Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.
- Bir taraftan kullanım, diğer taraftan üretim yapılır.

Artırımsal Geliştirme Süreç Modeli

Artırımsal Geliştirme Süreç Modeli



Artırımsal Geliştirme Süreç Modeli



Araştırma Tabanlı Süreç Modeli

- Yap-at prototipi olarak ta bilinir.
- Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.
- Yapılan 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.

Araştırma Tabanlı Süreç Modeli

Örnek

- En Hızlı Çalışan asal sayı test programı!
- En Büyük asal sayıyı bulma programı!
- Satranç programı!

Metodolojiler

- Metodoloji: Bir BT projesi ya da yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütünü.
- Bir metodoloji,
  - bir süreç modelini ve
  - belirli sayıda belirtim yöntemini içerir
- Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır

Bir Metodolojide Bulunması Gereken Temel Bileşenler (Özellikler)

- Ayrıntılandırılmış bir süreç modeli
- Ayrıntılı süreç tanımları
- İyi tanımlı üretim yöntemleri
- Süreçlerarası arayüz 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

Bir Metodolojide Bulunması Gereken Temel Bileşenler

- Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (IEEE, ISO, vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.
- Kullanılan süreç modelleri ve belirtim yöntemleri zaman içinde değiştiği için standart ve rehberler de sürekli güncellenmektedir.
- Bir kuruluşun kendi metodolojisini geliştirmesi oldukça kapsamlı, zaman alıcı ve uzmanlık gerektiren bir faaliyet olup, istatistikler yaklaşık 50 kişi/ay'lık bir iş gücü gerektirdiğini göstermektedir.

Bir Metodoloji Örneği

- Yourdan Yapısal Sistem Tasarımı Metodolojisi.
- Kolay uygulanabilir bir model olup, günümüzde oldukça yaygın olarak kullanılmaktadır.
- Çağlayan modelini temel almaktadır.
- Bir çok CASE aracı tarafından doğrudan desteklenmektedir.

Yourdon Yapısal Sistem Tasarım Metodolojisi

Aşama	Kullanılan Yöntem ve Araçlar	Ne için Kullanıldığı	Çıktı
Planlama	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Maliyet Kestirim Yöntemi, Proje Yönetim Araçları	Süreç İnceleme Kaynak Kestirimi Proje Yönetimi	Proje Planı
Analiz	Veri Akış Şemaları,Süreç Belirtimleri,Görüşme,Nesne ilişki şemaları ,Veri	Süreç Analizi Veri Analizi	Sistem Analiz Raporu
Analizden Tasarıma Geçiş	Akışa Dayalı Analiz, Süreç belirtimlerinin program tasarım diline dönüştürülmesi, Nesne ilişki şemalarının veri tablosuna dönüştürülmesi	Başlangıç Tasarımı Ayrıntılı Tasarım Başlangıç Veri tasarımı	Başlangıç Tasarım Raporu
Tasarım	Yapısal Şemalar, Program Tasarım Dili, Veri Tabanı Tabloları	Genel Tasarım Ayrıntılı Tasarım Veri Tasarımı	Sistem Tasarım Raporu

Süreç Modeli	Örnek Proje
1. Gelişigüzel Model	Öğrenci Ders Projeleri
2. Barok Model	Kullanım Kılavuzu Gerektiren Basit Projeler
3. Çağlayan Modeli	Hastane Sistemleri, Kontrol Sistemleri
4. V Modeli	CRM (Müşteri İlişkileri Yönetimi) Projeleri
5. Helezonik Model	Askeri Savunma Yazılım Sistemleri
6. Arttırımsal Geliştirme	Öğrenci Bitirme Projeleri
7. Prototipleme	Yüksek Maliyetli Projelerin İlk Aşamasının Bilgisayar Destekli Gerçekleştirilmesi
8. Evrimsel Geliştirme	Çok Birimli Organizasyonlar İçin (Bankacılık Uygulamaları)
9. Araştırma Tabanlı Model	En Hızlı Çalışan Asal Sayı Test Programı, En Büyük Asal Sayıyı Bulma Programı, Satranç Programı

YAZILIM ÖLÇÜTLERİ

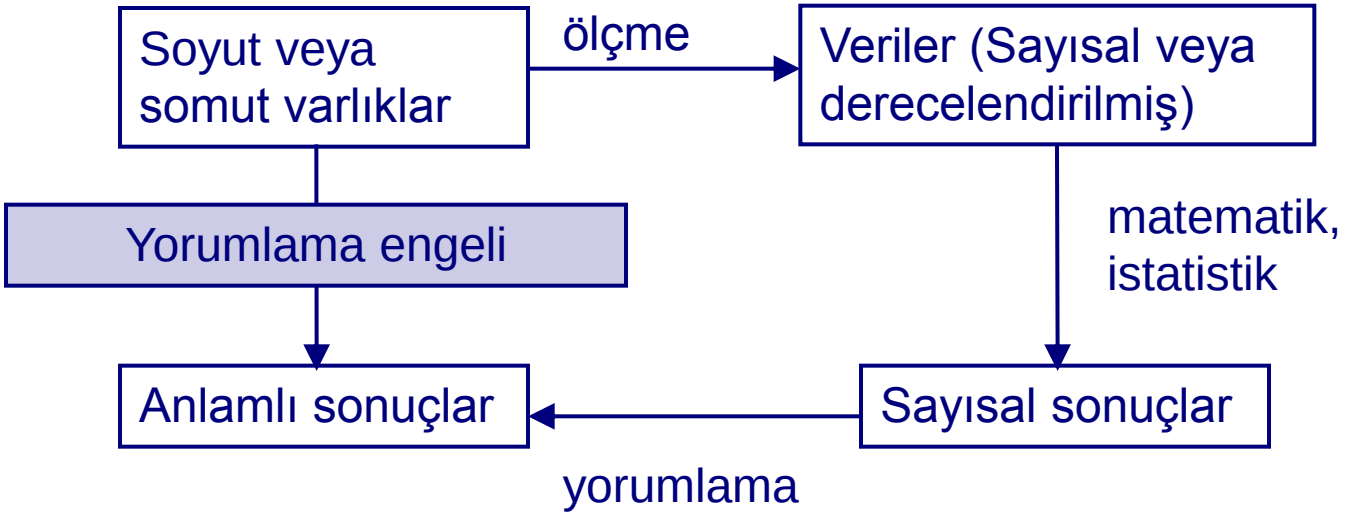
GENEL BİLGİLER

- Ölçme (Measuring): Somut veya soyut bir varlığın sahip olduğu bir özelliğini, sayısal veya derecelendirilmiş bir veri olarak ifade etmek.
  - Benim boyum 163 santimetredir.
  - Hava bugün 22 santigrat derecedir.
  - İlk ara sınav çok zordu.
- Ölçüt (Metric): Varlığın ölçülecek özelliğini ölçme biçimi.
  - Mesafe ölçütleri: Bir labirentteki Öklid ölçütü (Pisagor teoreminden) ve kuş uçuşu ölçütü.
  - Sıcaklık ölçütü: Santigrat ve Fahrenhayt
- Ölçüm (Measurement): Belli bir ölçüte göre yapılan ölçme eyleminin sonucu.
- Ölçme/ölçüt/ölçüm karışıklığı
  - İngilizce'de daha da zor
    - Measure –ment ve –ing son eklerini ben özellikle koydum
  - Türkçe'de daha kolay
    - Yine de neyin isim, neyin sıfat, neyin eylem olduğunu karıştırmamalı.
- Neden ölçeriz?
  - Gerçek dünya ile ilgili, işimize yarayacak, anlamlı sonuçlar elde etmek için.

YAZILIM ÖLÇÜTLERİ

GENEL BİLGİLER

- Yorumlama engeli (Intelligence barrier):
  - Ölçmenin sonucu, aradığımız sonuçları elde etmek için doğrudan bir yol sunmayabilir,
  - ya da yapacağımız yorumlama zor olabilir.
  - Örnek: Otostopçunun galaksi rehberi'nde hayatın anlamı: 42!



YAZILIM ÖLÇÜTLERİ

YAZILIM ÖLÇÜMÜ

- Yazılım ölçümü zordur:
  - Bir başka deyişle, yorumlama engeli yüksektir.
  - Zorluğun nedenleri:
    - Yazılımın karmaşıklığı
    - Ölçütlerin nicel doğası
- Yazılımı neden ölçeriz?
  - Ne kadar iyi bir ürün ortaya çıkardığımızı anlamak
  - Ne kadar iş yapacağımızı kestirmek
  - Böylece ne kadar zaman ve para harcayacağımızı anlamak
  - Ölçülemeyen ilerleme yönetilemez: Proje yönetiminde yazılım ölçütleri kullanılır.

YAZILIM ÖLÇÜTLERİ

YAZILIM KALİTE ÖLÇÜTLERİ

- Nicel kalite ölçütleri farklı kişilerce farklı şekillerde öbeklenmekte ve farklı dallara ayrılmaktadır.
- ISO 9126 kalite ölçütleri:
  - İşlevsellik
    - Uygunluk, doğruluk, güvenlik, …
  - Güvenilirlik
    - Olgunluk, hata bağışıklığı, …
  - Kullanılabilirlik
    - …
  - Verimlilik/Etkinlik
    - …
  - Bakım kolaylığı
    - …
  - Taşınabilirlik
    - …
- McCall ve arkadaşlarının kalite ölçütleri:
  - İşlevsel ölçütler
    - Doğruluk, Güvenilirlik, Bütünlük, Kullanılabilirlik, Verimlilik
  - Değiştirilme ölçütleri
    - Bakım kolaylığı, Esneklik, Sınanabilirlik
  - Taşınma ölçütleri
    - Taşınabilirlik, Yeniden Kullanılabilirlik, Birlikte Çalışabilirlik
- McConnell'a göre kalite ölçütleri:
  - İç kalite ölçütleri
  - Dış kalite ölçütleri

YAZILIM ÖLÇÜTLERİ

YAZILIM KALİTE ÖLÇÜTLERİ

- Dış kalite ölçütleri: Yazılımı kullananları ilgilendiren ölçütler.
  - Doğruluk(Correctness): Yazılımın hatalar içermemesi, gereksinimlerde belirtildiği şekilde çalışması.
  - Etkinlik(Efficiency): Bellek ve işlemci gibi sistem kaynaklarının en az oranda kullanımı.
  - Güvenilirlik(Reliability): Sistemin her koşulda istenildiği gibi çalışması, hatalar arasındaki ortalama zaman aralığının (MTBF) yüksek olması.
  - Güvenlik(Security): İzinsiz ve yetkisiz işlemler mümkün olmamalı.
  - Bütünlük(Integrity): Veriler ve işlemler arasındaki tutarlılığın korunması.
  - Uyarlanabilirlik(Adaptability): Sistemin değişik uygulamalar veya ortamlarda kullanılabilmesi için mümkün olduğunca az değişiklik gerektirmesi.
  - Hassaslık (Accuracy): Sistemin kendisinden beklenen işi mümkün olduğunca iyi yapabilmesi.
  - Sağlamlık(Robustness): Aykırı girişlere veya güç çalışma ortamlarına karşılık sistemin çalışmayı sürdürebilmesi.
  - Kullanılabilirlik(Usability): Yazılım kolay kullanılabilir olmalıdır.
  - …
- Bu ölçütler örtüşebilir, bazı durumlarda birbirinden daha iyi veya daha zor ayrılabilir.

YAZILIM ÖLÇÜTLERİ

YAZILIM KALİTE ÖLÇÜTLERİ

- İç kalite ölçütleri: Yazılımı geliştirenleri ilgilendiren ölçütler.
  - Yeniden kullanılabilirlik(Reusability): Sistemin parçalarının başka sistemlerde kullanılabilmesinin kolaylığı.
  - Bakım kolaylığı (Maintainability): Yazılıma yeni yetenekler eklemenin, yazılımdaki hataları gidermenin veya yazılımın başarımını attırmanın mümkün olduğunca kolay olması.
  - Esneklik(Flexibility): Yazılımın orijinal olarak tasarlandığı uygulamanın dışında çalışabilmesi için gerekli olan değişikliklerin olduğunca az olması.
  - Taşınabilirlik(Portability): Yazılımın farklı donanım ve işletim sistemleri gibi değişik çalışma ortamlarına kolaylıkla aktarılabilmesi.
  - Okunabilirlik(Readability): Kodun kaynak kodunun incelenmesinin kolay olması.
  - Anlaşılabilirlik(Understandablility): Yazılımın sistem, bileşen ve kod düzeylerinde anlaşılabilirliğinin mümkün olduğunca kolay olması. Okunabilirlik sadece kod düzeyinde anlaşılabilirliği sağlar.
  - Sınanabilirlik(Testability): Sistemin istenen gereksinimleri karşılayıp karşılamadığının sınanabilmesinin bileşen ve tüm sistem çapında mümkün olduğunca kolay olması.

YAZILIM ÖLÇÜTLERİ

ÖLÇME İLKELERİ

- Ölçme eyleminin içermesi gereken adımlar:
  - Tanımlama (Formulation): Ölçütler ölçülecek yazılıma uygun bir şekilde tanımlanır
    - Kullanılan yaklaşım: Yapısal programlama, NYP, vb.
    - Yazılımın türü: Gerçek zamanlı, gömülü, uygulama, vb.
  - Toplama (Collection): Tarif edilen ölçütlerin gerektirdiği verileri elde etme.
  - Hesaplama (Analysis): Ölçütlerin hesaplanması = Ölçümlerin elde edilmesi.
    - Matematiksel araçlar kullanılabilir.
    - Hesaplama mümkün olduğunca otomatik yapılmalıdır.
  - Yorumlama (Interpretion): Elde edilen ölçüm değerlerinden yararlı anlamlar çıkartılması.
  - Geri besleme/Kullanma (Feedback): Çıkartılan sonuçların yazılım ekibine bildirilmesi ve ekibin sonuçları kullanarak yazılımı iyileştirmesi.

YAZILIM ÖLÇÜTLERİ

ÖLÇME İLKELERİ

- Bir ölçütün sahip olması arzu edilen özellikler:
  - Uygun matematiksel özelliklere sahip olmalı:
    - Anlamlı bir ölçekte olmalı. Ör. 0-1 arası sonuçlar üretmeli.
    - Doğru (veya ters) orantıya sahip olmalı. Sonucun yükselmesi, ölçülen özelliğin iyi bir sonuca doğru ilerlemesi (gerilemesi) anlamına gelmeli.
  - Deneysel olarak doğrulanabilmeli
    - Doğrulanmasının ardından kullanılmalı.

YAZILIM ÖLÇÜTLERİ

ÖNERİLEN YAZILIM ÖLÇÜTLERİ

- Nesneye yönelik ölçütler:
  - Kaliteli bir yazılıma götüren tasarım ilkelerine yöneliktirler.
  - NYP'de çözümleme ve tasarım arasında kopukluk olmadığı için, aynı ölçütler çözümleme ve kodlama aşamalarında da kullanılabilir.
  - Böylece yazılım ekibi, 'kaliteli bir ürüne giden yolda' iz üstünde olup olmadıklarını anlayabilir.
  - Proje yöneticisi de, başka ölçütlerle birlikte, kestirimlerde bulunabilir.
- Chidamber ve Kemerer'in ölçütleri (CK metrics suite):
  - WMC: Sınıftaki ağırlıklı metot sayısı (Weighted Methods per Class).
  - DIT: Kalıtım ağacının derinliği (Depth of Inheritance Tree).
  - NOC: Alt sınıf sayısı (Number of Children)
  - RFC: Sınıfın yanıt kümesinin eleman sayısı (Response For a Class)
  - CBO: Sınıflar arası bağlaşım (Coupling Between Objects)
  - LCOM: Uyum eksikliği (Lack of COhesion in Methods)

YAZILIM ÖLÇÜTLERİ

CK ÖLÇÜTLERİ ÖRNEĞİ:

- WMC:
  - C1 sınıfının M1…Mn metotlarının karmaşıklıkları c1..cn.

$$WMC = \sum_{i=1}^{n} c_i$$

  - Eleştiriler:
    - Metot karmaşıklığı neye göre belirlenecek?
    - Belirlemedeki öznellik güçlü yön mü, zayıf yön mü?
  - Ölçütün rehberliği:
    - Bir sınıfın karmaşıklığını belirler.
    - Çok sayıda metodu olan sınıf:
      - Çok fazla sorumluluk yüklenmiştir, dağıtılması uygun olabilir.
      - Yüksek uyumun olup olmadığına tekrar bakılmalıdır.
      - Uygulamaya özeldir, yeniden kullanılabilirliği düşüktür.

YAZILIM ÖLÇÜTLERİ

DİĞER KALİTE ÖZELLİKLERİNE YÖNELİK ÖLÇÜTLER

- Bazı bağlaşım ölçütleri
  - COMIAS
  - CBMC
- Sınanabilirlik ölçütleri
  - Halstead ölçütleri (Tartışmalı)
  - Binder'in seçtiği ölçütler
- Bakım kolaylığı ölçütleri
  - IEEE Std. 982.1-1998'de yazılım olgunluk ölçütü

NE YAPILABİLİR?

- Ölçütlerin büyük çoğunluğu mükemmel değildir.
- Yine de bu zayıf noktalar genellikle çok özel durumlarda ortaya çıkar.
- Bu nedenle ölçütler kullanılmalı, ancak tabulaştırılmamalı, sadece (çok da hassas olmayan) bir rehber olarak kullanılmalı.
- Belli bir kalite ölçütüne yönelik olarak, şimdiye dek önerilen ölçütlerden bazıları seçilip, sezgisel olarak bir araya getirildikten sonra piyasada sınanarak iyileştirilebilir.
  - Sezgisel yetenek nasıl bulunacak?
  - Özel sektör gerekli çabaya nasıl ikna edilecek?

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yöneticisi:
  - Teknik ekibin bir parçası olduğundan teknik yetenekleri yüksek olmalıdır.
  - Ağırlıklı olarak insanlarla ilgili eylemlerde bulunacağından, sosyal ve yönetimsel yetenekleri de yüksek olmalıdır.
- İyi bir teknik yöneticinin özellikleri:
  - Teknik ekibi istekli kılabilmelidir.
  - Kişileri ve yazılım geliştirme sürecini, üzerinde çalışılan ürüne/ürün parçasına göre düzenleyebilmelidir.
    - Düzenleme küçük veya büyük ölçekte olabilir.
  - Ürün ve süreç sınırları belli olsa da, ekibini yaratıcı fikirler üretmeye teşvik edebilmelidir.
  - İyi bir sorun çözücü olmalıdır.
    - Hem teknik hem de yönetimsel sorunlarla uğraşabilmelidir.
    - Sorunlara tanı koyabilmeli ve ortaya uygun bir çözüm koyabilmelidir.
    - Seçilen çözüm tıkandığında ısrarcı olmamalıdır.
  - Sorumluluk alabilmelidir.

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Yazılım geliştirme ekibi (teknik ekip):
  - Takım ruhuna uygun kişilerden oluşmalıdır:
    - Takım üyeleri birbirine saygı duymalıdır.
    - Takım üyeleri ortak amaç etrafında kenetlenebilmelidir.
    - Takım üyeleri birbirlerini tamamlayan yeteneklere sahip olmalıdır.
  - Takım ruhunu bozan etkenler:
    - Telaşlı iş ortamı.
    - Sık ortaya çıkan hayal kırıklıkları ve başarısızlıkların takım üyeleri arasındaki sürtüşmeyi arttırması.
    - Doğru yönetilemeyen yazılım geliştirme süreci.
    - Takım yapısının ve rollerinin belirsiz tanımlanması.

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yapıları:
  - Kapalı yaklaşım:
    - Geleneksel bir yetki hiyerarşisi ve kontrol mekanizmaları bulunur.
    - Geçmiş deneyimlere benzer projelerde başarılı bir yapıdır.
    - Yaratıcı fikirler ortaya çıkarmak için çok uygun değildir.
  - Rastgele (Random) yaklaşım: Serbest yaklaşım.
    - Takım üyelerinin bireysel ve teknik yeteneklerine göre kendi aralarında bir yapı kurmasıdır.
    - Yaratıcı fikirler ortaya çıkarmak için en uygun yaklaşımdır.
    - Disiplin elde etmek zor olabilir.
  - Açık yaklaşım: Kapalı ve rastgele arasında.
    - Kontrol mekanizmaları bulunur ancak yapılanma serbesttir.
    - Demokratik yapı.
    - Karmaşık sorunların çözümü için uygun.
    - Etkinliği (efficiency) sağlamak zor olabilir.

PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yapıları (devam)
  - Eşzamanlı (synchronous) yaklaşım:
    - Problemin takımın üzerine düşen bölümünün de alt parçalara ayrılabildiği durumlarda kullanılabilir.
    - Takım kendi içerisinde problemin alt parçalarını paylaşır.
    - Alt takımlar arasında etkileşim azdır.
- Takım içi ve takımlar arası haberleşme:
  - Resmi yollar: Yazı ile, zamanlı mesajlaşma ile, kurallı ve zamanlanmış toplantılar ile.
  - Gayrı resmi yollar: Sözlü iletişim, kişisel etkileşimler, gün içerisinde gerektikçe.

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE ÖLÇÜM

- Proje Ölçümü:
  - Yazılımın ölçülmesidir (İncelendi).
  - Odak: Teknik düzey.
  - Amaç: İç kalite ölçütlerini yüksek tutmak
  - Yöntem: Ölçüm sonuçlarına göre, yazılım geliştirme ekibini iyiye doğru yönlendirmek.
- Süreç Ölçümü:
  - Yazılım geliştirme sürecinin ölçülmesidir.
  - Odak: Yönetimsel düzey.
  - Amaç: Dış kalite ölçütlerine yöneliktir.
  - Yöntem: Sürecin tüm aşamalarında tutulan istatistiklere göre, hem teknik hem de yönetimsel açıdan süreçleri iyileştirmek.

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE ÖLÇÜM

- Kalite ile ilgili etkenler:
  - Kişiler: Çalışanların yetenek ve istekliliği en önemli etkendir.
  - Ürün: Ürünün karmaşıklığı kaliteye olumsuz etki eder. Karmaşıklıkla artan zorluk düzeyi, takımın maneviyatını kırabilir.
  - Teknoloji: Yazılım geliştirme araçları ve donanım bileşenlerinin kalitesi.
    - Kalite: Olgunluk, yeterlilik, etkinlik, eğitimsel belgeler, vb.
  - Çevresel koşullar: Ana etkenlerle de ilişkilendirilebilir.
    - Müşterinin özellikleri: İletişim yeteneği, işbirliği yeteneği, vb.
    - İş koşulları: İş akışı kuralları, şirket kültürü, vb.
    - vb.
- Ölçüt toplama ve gizlilik düzeyleri:
  - Bireysel düzey: Kişiye özel olmalıdır.
  - Takım düzeyi: Takıma özel olmalıdır.
  - Proje düzeyi: Tüm çalışanlara özel olmalıdır.
- Gizlilik düzeyi tartışması:
  - Üstün, astlarının ölçütlerini izleyebilmesi ancak denklerin birbirlerinin ölçütlerini görememesi ise bir başka yaklaşımdır.
  - Amaç insanları utandırmak, övmek veya tehdit etmek olmamalı.

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE ÖLÇÜM

- Süreç Ölçütleri:
  - Hata sayısı: Bulunan hata sayısı.
    - Bakım aşamasından önce ve sonra bulunan hataların sayısı, ayrı öneme sahiptir.
    - Hata giderme etkinliği:
      - HGE = TÖ / (TÖ + TS)
      - TÖ: Ürünün müşteriye tesliminden önce bulunan hata sayısı
      - TS: Teslimden sonra (bakım aşamasında) bulunan hata sayısı
      - TÖ ve TS yerine, yazılım geliştirme sürecinin ardışıl adımları olan A(i) ve A(i+1) kullanılabilir. Örneğin:
      - A(i) : Çözümlemede bulunan hata sayısı
      - A(i+1) : Tasarımda bulunan çözümleme kaynaklı hata sayısı.
  - Doğruluk: Bulunan hata sayısının proje boyutuna oranı.
  - Bakım kolaylığı: Bir hatanın bulunması ile giderilmesi arasında geçen zaman.
  - Güvenlik: Bulunan güvenlik açıkları, açıkların ciddilik düzeyi, vb. gibi veriler üzerinde yapılan ölçümler.
  - Ve diğer dış kalite ölçütleri…

YAZILIM PROJE YÖNETİMİ

PROJE YÖNETİMİNDE ÖLÇÜM

- Süreç ölçütlerinin kullanımı:
  - Ölçütler amaç değil, araç olmalı.
  - Şirketin boyutu ile ölçüm yapmaya ayrılan çaba orantılı olmalı.
  - Öncelikleri ve hedefleri belirleyip ona uygun ölçütler kullanılmalı.
    - Amaca yönelik ölçümler yapılmalı.
  - SEI (Software Engineering Institute) ve SPC (Software Productivity Center)'nin önerdiği ölçüt geliştirme yaklaşımları mevcuttur.
    - Ve daha başkaları…
  - Süreç iyileştirme modelleri çeşitli ölçütlerin kullanılmasını da gerektirir.

YAZILIM PROJE YÖNETİMİ

YAZILIM PROJELERİNİN VAZGEÇİLMEZ ARAÇLARI

- IDE'ler.
- UML modelleme araçları.
  - İki yönlü dönüşüm yeteneğine sahip olması (model ve kod arasında) tercih edilir.
- Sürümlendirme yazılımı (version control systems)
- Sınama yazılımı (testing framework)
  - Bir yapılandırma yazılımı (build system) ile tümleşik olması tercih edilir.
- İş kalemleri izleme yazılımı (work item tracking)
- Tüm araçların IDE tümleşik olması tercih edilir.
  - Java: IBM Rational Application Developer
  - .NET: Microsoft Team System

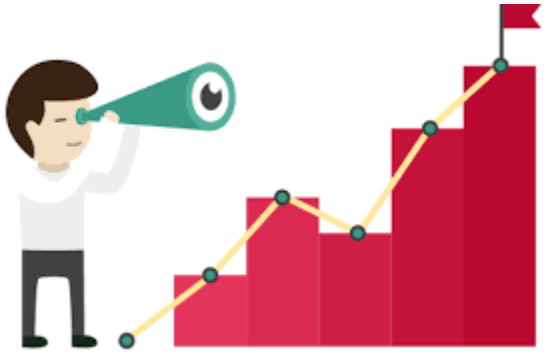
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

5.Hafta Sistem Çözümleme

Dr. Muhammet BAYKARA

Bölüm Hedefi

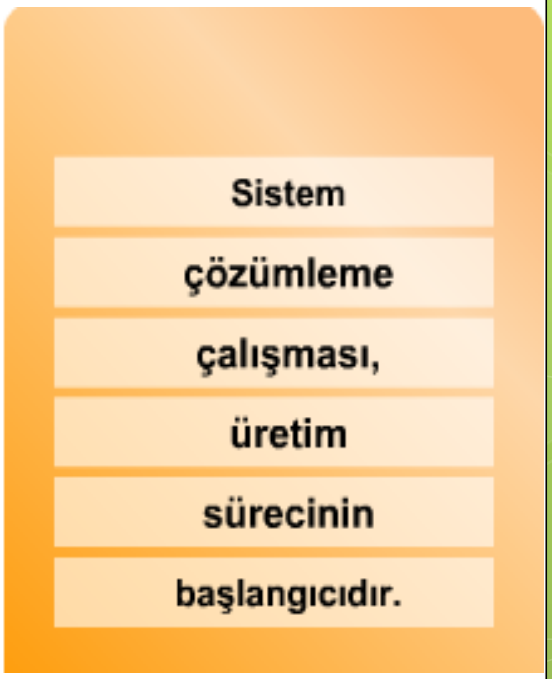
- Bu bölümde, yazılım geliştirme aşamalarından kullanıcı kesimini en çok ilgilendiren aşama olan çözümleme aşamasında kullanılan yöntem ve araçlar olan Kullanıcı arayüzü prototipleme, veri modelleme, süreç işlem modelleme ve veri toplama yöntemleri açıklanmaktadır.



Sistem Çözümlemeye Giriş

- Sistem çözümleme çalışması, üretim sürecinin başlangıcıdır. Bu aşamada temel olarak mevcut sistemin nasıl çalıştığı araştırılır.

- Mevcut sistemin incelenmesi sırasında temel hedef gereksinimlerin saptanmasıdır. Bu işlemden sonra önerilen sistem için modelleme yapılır. Söz konusu model, mantıksal model olarak bilinir.



Sistem Çözümleme

- Çözümleme çalışmasında mutlaka bir model/yöntem kullanma zorunluluğu vardır. Aksi durumda, çalışma dağınık biçimde sürer, denetlenemez ve başarısız olur.
- Bu bölümde, yapısal sistem geliştirme yaklaşımında kullanılan yöntemlerden en yaygın olanlarına örnekler verilmiştir. Yöntemler, veri modelleme ve süreç modelleme yöntemleri olarak iki başlık altında incelenmiştir.

Gereksinim Nedir?

- Bir sistem geliştirilirken, kullanıcının sistemin işlevleri ile ilgili beklentileri sistemin amaçlarını oluşturur. Gereksinim, sistemin amaçlarını yerine getirme yeteneği olan bir özellik ya da belirtim olarak tanımlanmaktadır.
- Bir kuruluş için personel bordro sistemi geliştirdiğimizi varsayalım. Maaş bildirim formlarının aylık olarak hazırlanması, kuruluşun değişik birimlerinden bu bilgilere erişim istekleri, gereksinimlere örnektir.

Gereksinim Türleri

İşlevsel Gereksinimler

Sistem ile çevresi arasındaki iletişimi belirleyen gereksinimler işlevsel gereksinim olarak tanımlanır. İşlevsel gereksinimler ayrıca sistemin herhangi bir durum karşısındaki davranışını belirler. Maaş çeki hazırlama örneğine dönecek olursak, işlevsel gereksinimler; maaş çekinin ne zaman hazırlanacağı, çek hazırlamak için ne tür girdiler verilmesi gerektiği, hangi koşullarda bir kişiye maaş verilmeyeceği gibi gereksinimler işlevsel gereksinimlerdir.

İşlevsel Olmayan Gereksinimler

Kullanıcının sorunundan bağımsız olarak çözülmesi gereken sorunlar işlevsel olmayan gereksinim olarak tanımlanır. Bir çok kaynakta bu terim yerine "Sistem Kısıtları" terimi de kullanılmaktadır. Kullanılacak bilgisayarın türü, yazılım geliştirme ortamı, kullanılacak veri tabanı yönetim sistemi vb. bu tür gereksinimlere örnek olarak verilebilir.

Tekrar

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Fiziksel Çevre

- İşlevlerin geliştirileceği, işletileceği aygıtlar nerededir?
- Sistem tek bir yerde mi olacak, birden çok, fiziksel olarak birbirinden ayrılmış yerler söz konusu mu?
- Sıcaklık, nem oranı veya manyetik etkileşim gibi çevresel kısıtlamalar var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Arayüzler

- Girdiler bir mi, yoksa birden çok sistemden mi geliyor?
- Çıktılar bir mi, yoksa birden çok sisteme mi gidiyor?
- Verinin nasıl biçimlendirileceğine ilişkin önerilen bir yol var mı?
- Verinin kullanılacağı, önerilen bir ortam var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Kullanıcı ve İnsan Etmeni

- Sistemi kim kullanacak?
- Çeşitli tipte kullanıcılar olacak mı?
- Her bir kullanıcı tipinin yetenek düzeyi nedir?
- Her kullanıcı tipi için ne tür bir eğitim gereklidir?
- Bir kullanıcının sistemi anlaması ne denli kolay olacaktır?
- Bir kullanıcının sistemi kötü amaclı kullanabilmesi ne ölçüde zordur?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

İşlevsellik

- Sistem ne yapacak?
- Sistem bunu ne zaman gerçekleştirecek?
- Sistem nasıl ve ne zaman değiştirilebilir veya güçlendirilebilir?
- Çalışma hızı, yanıt süresi ya da çıktı üzerinde kısıtlayıcı etmenler var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Belgeleme

- Ne kadar belgeleme gereklidir?
- Belgeleme hangi kullanıcı kitlesini hedeflemektedir?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Veri

- Hem giriş hem çıkış için verinin biçimi ne olmalıdır?
- Bu veri ne sıklıkta alınacak veya gönderilecektir?
- Bu veri kesinlik (doğruluk) ölçüsü ne kadar olmalıdır?
- Hesaplamalar hangi duyarlık derecesine kadar yapılacaktır?
- Sistemden ne kadar veri akışı olacaktır?
- Belli bir zaman suresince veri kaynağında saklanacak mıdır?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Kaynaklar

- Sistemi kurmak, kullanmak ve bakımını yapmak için ne kadar malzeme, personel ve diğer kaynaklara ihtiyaç vardır?
- Geliştiriciler hangi yeteneklere sahip olmalıdır?
- Sistem ne kadar bir fiziksel yer kaplayacaktır?
- Güç (besleme), ısıtma veya soğutma için gereksinimler nelerdir?
- Geliştirim için tavsiye edilen bir zaman çizelgesi var mıdır?
- Geliştirme veya yazılım ve donanım icin harcanacak para üzerinde bir sınırlama var mıdır?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Güvenlik

- Sisteme veya bilgiye erişim denetlenmeli midir?
- Bir kullanıcının verisi diğerlerininkinden nasıl
- ayrıştırılacaktır?
  Kullanıcı programları diğer program ve işletim
- sistemlerinden nasıl ayrı tutulacaktır?
- Sistem hangi sıklıkta yedeklenecektir?
- Yedek kopyaları başka bir yerde mi saklanmaktadır?
  Yangın ve hırsızlığa karşı ne tür önlem alınacaktır?

Gereksinimler

- Fiziksel Çevre
- Arayüzler
- Kullanıcı ve İnsan Etmeni
- İşlevsellik
- Belgeleme
- Veri
- Kaynaklar
- Güvenlik
- Kalite Güvencesi

Kalite Güvencesi

- Güvenirlik için gereksinimler nelerdir?
- Sistemin özellikleri, insanlara nasıl aktarılmalıdır?
- Sistem, hataları kendisi bulup gidermeli midir?
- Sistem çökmeleri arasında öngörülen zaman nedir?
- Bir çöküm sonrasında sistemi tekrar başlatmak için uygun görülebilecek bir süre var mıdır?
- Sistem, tasarımda yapılacak değişiklikleri nasıl özümseyecektir?
- Bakımdan anlaşılan sadece hata gidermek midir, yoksa sistemi geliştirmeyi de içerir mi?
- Kaynak kullanımı ve yanıt süresine ilişkin verimlilik ölçütleri nedir?
- Sistemi bir yerden bir yere veya bilgisayardan diğerine taşımak ne kadar kolaylıkta olmalıdır?

Tekrar

Gereksinim Özellikleri

- Gereksinimler sadece bir sisteme ve sistemden bilgi akışını ve sistemce yapılan veri dönüşümünü tanımlamaz, aynı zamanda sistemin performansı üzerindeki kısıtları da tanımlar.
- Bu yüzden, gereksinimler üç amaca hizmet eder:
- Birincisi; geliştiricilerin, müşterilerin sistemin nasıl çalışmasını istediklerini anlamalarını sağlar.
- Gereksinimler, ayrıca tasarımcılara, sonuç sistemin ne işlevsellik ve özellikte olacağını söyler.
- Üçüncüsü; gereksinimler, sınama ekibine, kullanıcıyı, sunulan sistemin istenen sistem olduğuna ikna etmek için neler göstermeleri gerektiğini söyler.

Hem geliştiriciler hem de kullanıcılar gereksinimleri kullandığı için, gereksinimlerin geçerliliğinin doğrulanması gerekir. Doğrulama süreci aşağıda belirtilen yedi kriteri içerir:

1. **Gereksinimler doğru oluşturulmuş mudur?** Hem geliştirici, hemde kulanıcı tarafından belirsizliği giderilmiş, hatasız olarak onaylanmış olduğunu güvenceye almak için gözden geçiriniz.
2. **Gereksinimler tutarlı mı?** Yani, hiç çelişen, çift anlamlı, anlaşılmayan gereksinimler var mı?
3. **Gereksinimler tam mı?** Eğer tüm olası durumlar bir gereksinimle açıklanıyorsa, gereksinimler tamdırlar. Bu nedenle, örneğin bir çalışan ücretsiz izne çıktığında, birisi terfi ettiğinde veya bir kişi avans istediğinde, ne olacağı, örneğin bir PERSONEL sistemi tarafından açıklanmalıdır. Tanımlama, kullanıcı tarafından istenen tüm özellikleri içeriyorsa, bir sistem için "dışsal tam" dır denir. Bir tanımlama; eğer gereksinimler içersinde açıklanmamış referanslar yoksa "içsel tam" olarak tanımlanır.

Gereksinimlerin Geçerliliğinin Doğrulanması Kriterleri

4 **Gereksinimler gerçekçi mi?** Kullanıcının sistemden istediği gerçekten yapılabilir mi? Geliştirme zamanının uzun olduğu durumlarda, teknolojik değişime uyulabilecek biçimde tanımlı mı?

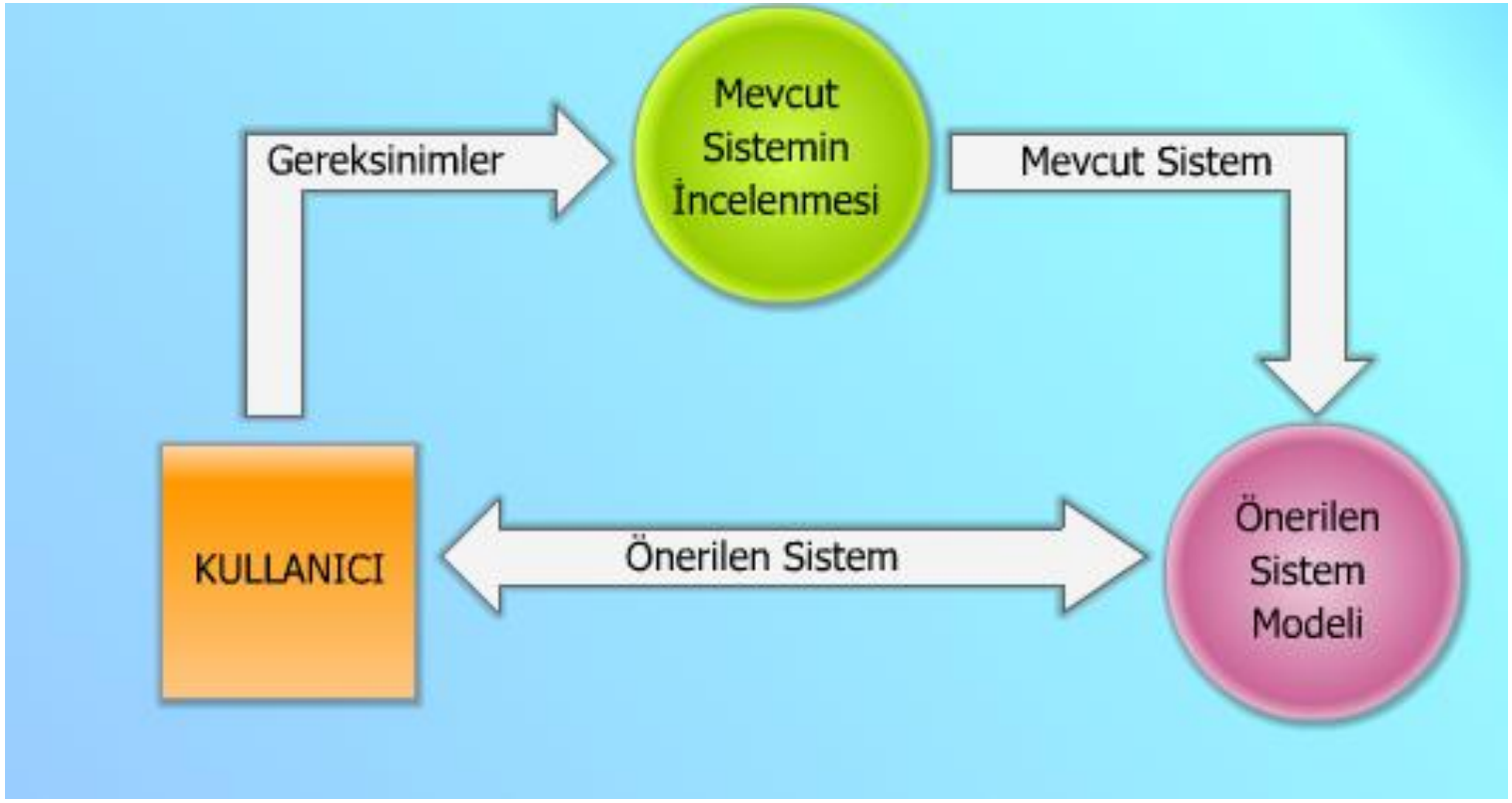
5 **Her gereksinim kullanıcı tarafından istenen bir şeyi mi tanımlamaktadır?** Bazı durumlarda bir gereksinim bizi gereksiz yere kısıtlayabilir. Örneğin, kullanıcı, ünü iyi olduğundan dolayı, belirli bir XYZ bilgisayarının kullanılmasını isteyebilir. Öte yandan, XYZ, istenen sistemin gerçekleştirmesi için en uygun bilgisayar olmayabilir. Bu tür gereksinimler doğrudan sistemin hedeflerini adreslemez. Yalnızca kullanıcının sorununu çözmek için doğrudan işe yarayacak gereksinimler üzerinde yoğunlaşılmalıdır.

Gereksinimlerin Geçerliliğinin Doğrulanması Kriterleri

6 **Gereksinimler doğrulanabilir mi?** Gereksinimlerin karşılandığını gösterebilmek için sınama senaryoları oluşturulabilir mi? Ya da doğrudan sınama yapılabilir mi?. Kullanıcı, bir hız tanımlaması yapmadan ve yanıt süresinin koşullarını tanımlamadan 'hızlı yanıt süresi' isteyebilir. Örneğin, 'hızlı yanıt süresi', daha doğrulanabilir olarak şu şekilde tekrar tanımlanabilir; 'sistemde kelime işlem programını kullanan en çok (örneğin: 32) sayıda kullanıcı ile, sistem bir kullanıcının ekranına bir belgeyi bir sonraki sayfasıyla birlikte en çok 5 saniyede getirmelidir.'

7 **Gereksinimler izlenebilir mi?** Her sistem işlevi, kendisini zorunlu kılan bir dizi gereksinim olarak izlenebilir mi? Sistemin özel bir yönüyle ilgili bir dizi gereksinim bulmak kolay mı? Örneğin, tüm iletişim gereksinimlerini gözden geçirmek için bütün gereksinimler okunmak zorunda mı?

Gereksinim Çözümleme Çalışması



Gereksinim Çözümleme Çalışması

- Geliştirilecek bilgi sistemi ya da yazılımla ilgili olarak tüm gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve bir gösterim biçimi ile açıklanması (modellenmesi) çalışmaları Sistem Çözümleme Çalışması olarak adlandırılır.

Temel olarak çözümleme çalışması:

- **Mevcut Sistemin İncelenmesi**
- **Önerilen Sistemin Modellemesi**

Olmak üzere iki aşamalı yapılır. Öncelikle mevcut sistem incelenir bu incelemenin tamamlanmasından sonra önerilen yapı modellenir.

Mevcut Sistemin İncelemesi

- Bu çalışmada temel amaç yazılım gelişitirilecek olan sistemin anlaşılması ve tanınmasıdır. Bu amaçla görüşme yapma, gerekirse anket yapma yöntemleri kullanılır.
- Yapılacak görüşmelerde, mevcut sistemde elle yürütülen tüm işlemler, girdi, işlev, çıktı ve diğer işlevlerle olan ilişkiler bazında sorgulanır. İlgili yönerge, kanun ve yöntemler kullanıcıdan edinilir.

Mevcut Sistemin İncelemesi

- Sorgulama ya da inceleme işleminin belirtilmesi amacıyla çeşitli yöntemler kullanılır. Sorgulama sonucu elde edilen bulgularla ilgili olarak kullanıcıdan geri bildirim alınır.
- Mevcut sistemde elle yürütülen işlerde, kullanılan form, defter ve yazışma örnekleri, sistemin veri boyutunu oluşturur. Bu tür bilgilerin birer kopyaları bir sonraki aşamada veri modellemesi için edinilir.

Önerilen Sistemin Modellemesi - Mantıksal Tasarım

- Mevcut sistemin modellenmesinden sonra, bilgisayarlı ortamda işlerin yapılabilmesi amacıyla önerilecek sistemin modeli oluşturulur. Bu model, önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünü içerir.
- Bu modelleme daha çok, bilgi sistemini geliştirecek teknik personele (sistem tasarımcıları, programcılar) yöneliktir. Bu model aynı zamanda '**mantıksal model**' olarak ta tanımlanır.

Önerilen Sistemin Modellemesi - Mantıksal Tasarım

- Mantıksal model, önerilen sistemin veri yapısını ve süreç yapısını hem genel hem de ayrıntılı olarak tanımlar. Mantıksal Model, kolaylıkla fiziksel modele (Program parçaları, veri tabanı tabloları vb) dönüştürülebilir bir yapıdadır.

- Bazı yazarlar, '**mantıksal model**' üretimini tasarım aşamasının bir parçası olarak ele alırlar ve konuyu '**tasarım**' başlığı altında incelerler.

Gereksinim Verisi Toplama

- Mevcut sistemin incelenmesi sırasında kullanılabilecek temel yöntemler:

  1. Sorma
  2. Psikolojik Türetme Teknikleri
  3. İstatistiksel Teknikler

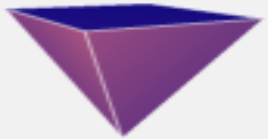
biçiminde özetlenebilir.

Sorma Yöntemi

- Sorma yöntemi, gereksinim verilerinin toplanması sırasında kullanılan en önemli yöntemlerden biridir.
- Karşılıklı görüşme ya da anket yolu ile uygulanır.

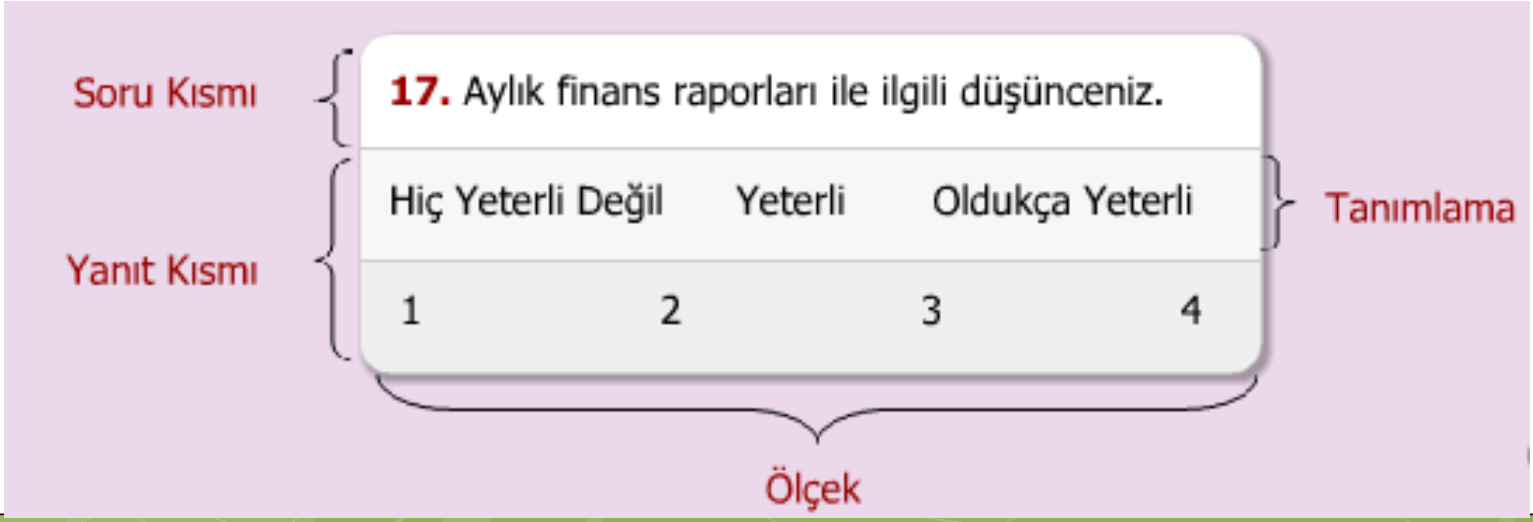
1. Karşılıklı Görüşme

- Karşılıklı görüşme sırasında, gereksinimleri ilişkin amaçlar, düşünceler, resmi olmayan yöntemler, duygular ve düşünceler araştırılır.
- Karşılıklı görüşme, veri toplama için en etkin yollardan biridir.

Piramit Tarzı	**Piramit Tarzı:** Özel sorularla başlayıp, giderek genel sorularla sürdürme.
 Koni Tarzı	**Koni Tarzı:** Genel sorularla başlayıp özel sorularla sürdürme.
Elmas Tarzı	**Elmas Tarzı:** Özel sorularla başlayıp, genel sorularla sürdürme, tekrar özel sorularla sona erdirme.

2. Anket

- Anket yöntemi, bir başka veri toplama yöntemidir. Kullanıcı sayısının fazla olduğu durumlarda, eğilimleri ve davranış biçimlerini saptamak amacıyla yapılır.
- Genelde yazılı test biçiminde hazırlanır. Bir anket sorusu temel olarak soru kısmı ve yanıt kısmı olmak üzere iki kısımdan oluşur.



Psikolojik Türetme Teknikleri

- Özellikle, belirsizliğin fazla olduğu ve zayıf yapılı ortamlarda, bilgi edinebilmek amacıyla insan psikolojisine dayalı teknikler kullanılır. Bu teknikler temelde görüşme ve ankete dayalı tekniklerdir. Diğerlerinden farkı, bilgi üretmek için psikolojide bilinen "üçleme" tekniğini kullanmasıdır.

- Bazı diğer psikolojik türetme yöntemleri, karar verme ortamlarında bilgi gereksinimlerini saptamak amacıyla algılama haritaları ve neden-etki çizgeleri kullanmaktadır.

İstatistiksel Teknikler

- Veri yoğun ve veri hacmi yüksek olan ortamlarda, verinin özelliklerini belirlemek amacıyla istatistiksel teknikler kullanılır.
- Bu yöntemlerden en çok bilinen ikisi **Örnekleme Yöntemi** ve **PIRA Modeli**dir.

İstatistiksel Teknikler

- **Örnekleme yöntemi**, bir topluluk içerisinden, sistematik yolla temsil edici örnek alma biçiminde tanımlanır. Amaç, veri toplama hızınını arttırmak ve verilerdeki çelişkileri önlemektir.
- Değişik örneklem türleri kullanılabilir; Basit gelişigüzel örneklem, kamaşık gelişigüzel örneklem, amaçlı örneklem vb. Örneklem boyutu, belirli bir güvenirlik düzeyinde olmak koşulu ile yine istatistiksel teknikler kullanılarak belirlenir.

İstatistiksel Teknikler

- **PIRA (Personal, Interactive, Report and Analysis) Modeli**, bilgi gereksinimlerinin tanımlarını belirli normlara bağlı olarak açıklamayı hedefler. Kişilerin bilgiye dayalı tercihlerini belirlemek amacıyla kulanılır.

Veri Modelleme Yöntemleri

- Önerilen sistemin mantıksal modelinde veri yapısını açıklamak amacıyla 'Veri Modelleme' yöntemleri kullanılmaktadır. Bu yöntemler, veri yapısını çeşitli düzeylerde tanımlama (en soyut düzeyden en ayrıntı düzeye kadar) amacını güder. Sistem Çözümleme aşamasında en yaygın olarak kullanılan veri modelleme yöntemleri:

- Nesne İlişki Şemaları
- Veri Sözlüğü

olarak bilinmektedir.

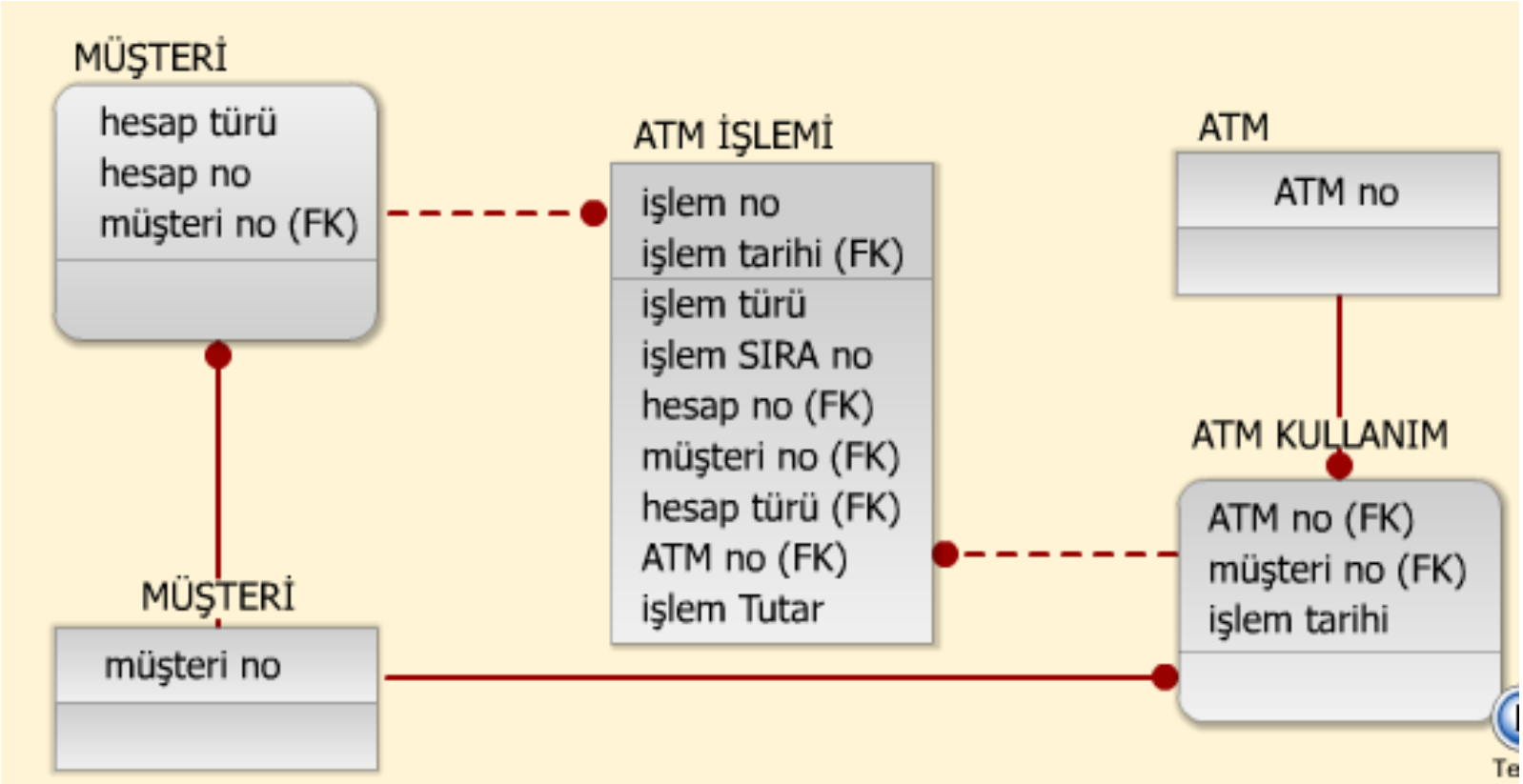
Nesne ilişki şemaları

- Bir veri nesnesi, üç temel özelliği ile bilinir:

  1. Veri nesnesi varlığının adı: Veri nesnesi varlığını tanımlayan özelliktir.
  2. Veri nesnesi varlığının özellikleri
  3. Veri nesnesi varlığının diğer veri nesnesi varlıklarına referansı: Veri nesnesi varlığının diğer veri nesneleri ile olan ilişkisinin belirtilmesi amacıyla kullanılır. Bu amaçla her bir veri nesnesini tek olarak belirleyen bir belirteç (anahtar) kullanılır. Söz konusu belirteç veri nesnesi varlığının ad özellikleri arasında yer alır.

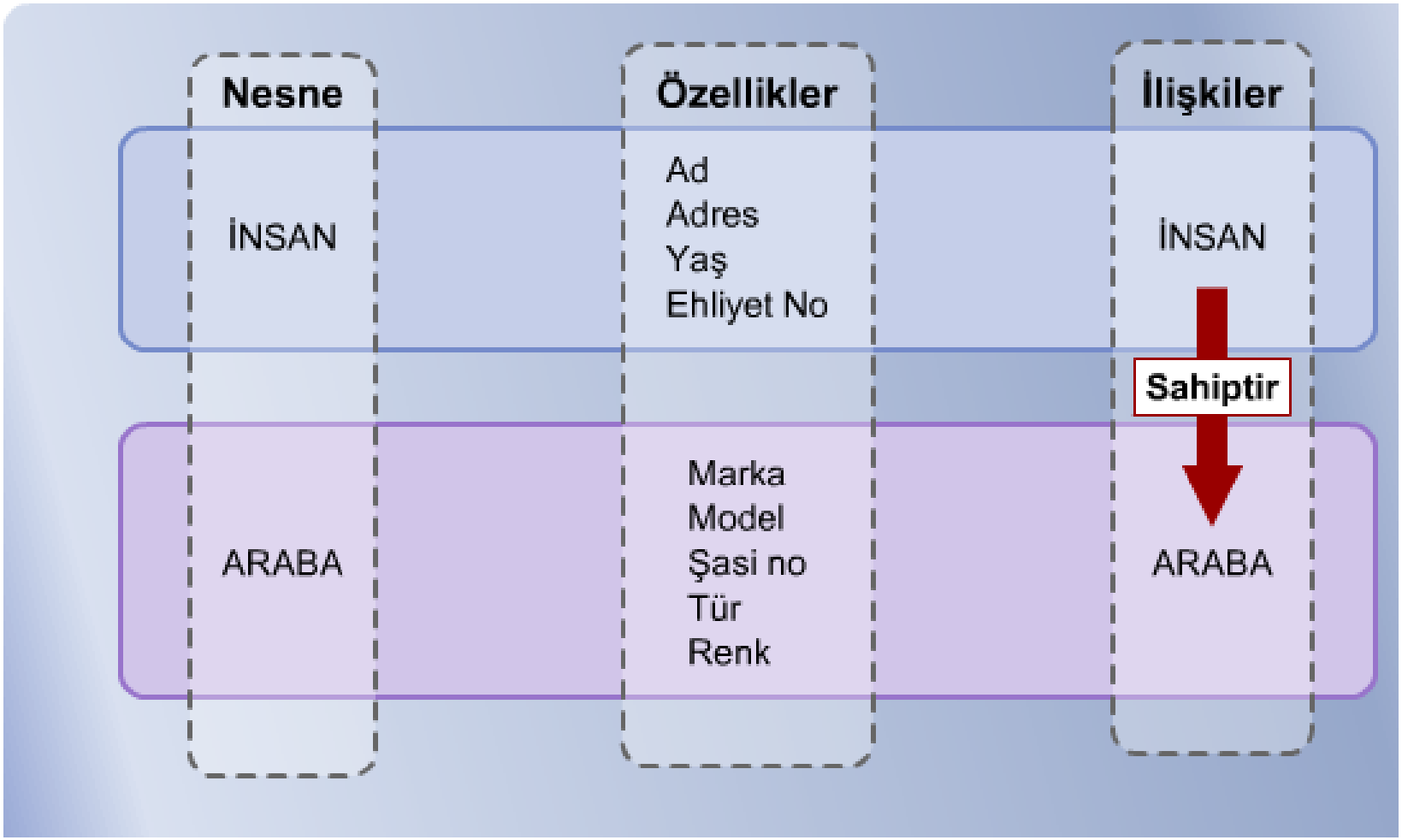
Nesne-İlişki Şemaları

- Geliştirilecek sistemin kullanacağı ana veri nesneleri ve aralarındaki ilişkileri belirtir.



Örnek - 1

ARABA ve **İNSAN** adlı iki veri nesnesi düşünelim.

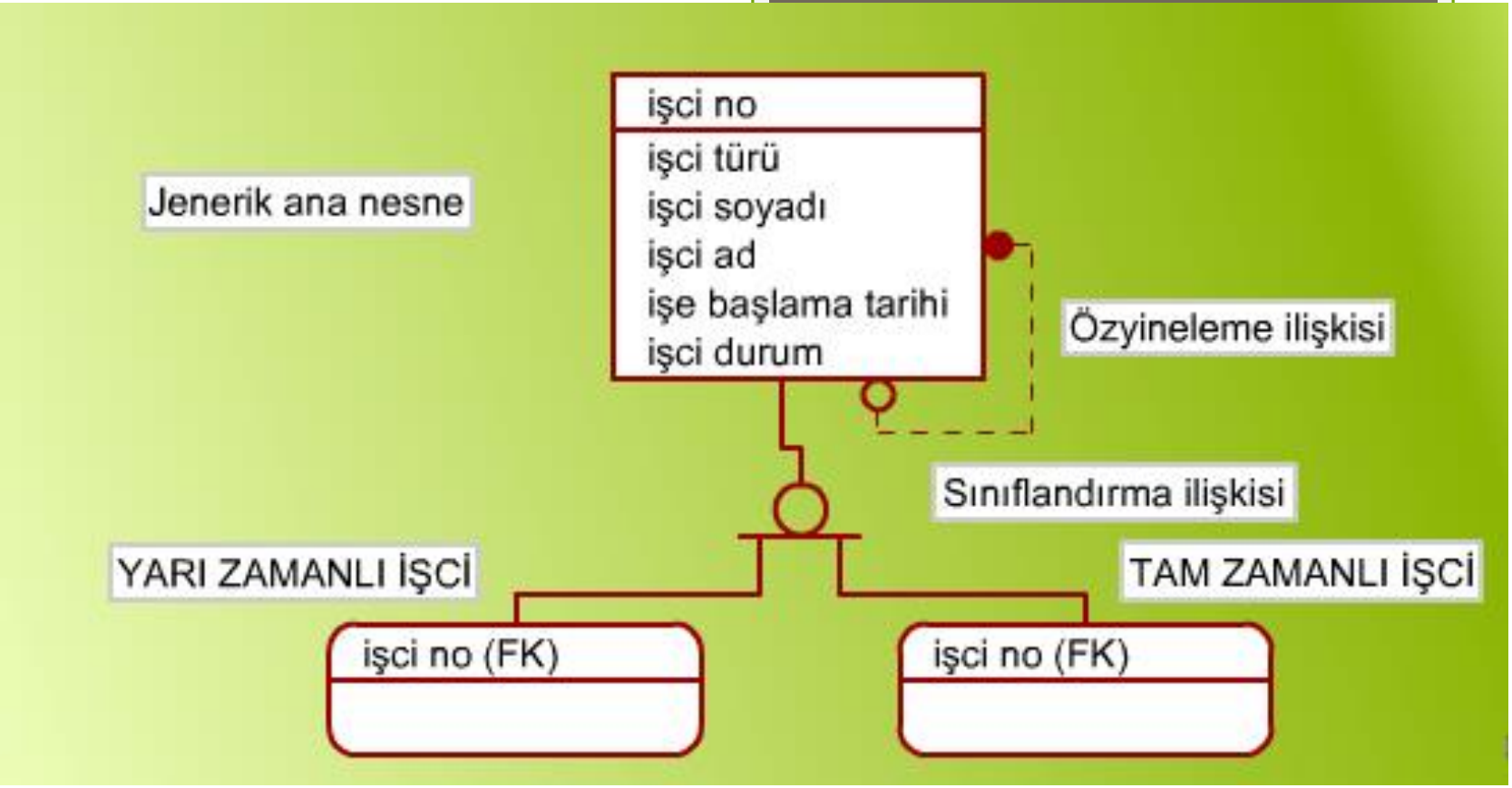


Yukarıda **ARABA** ve **İNSAN** nesnelerinin özellikleri verilmektedir. Araba nesnesinin varlıklarına ilişkin örnekler is

Marka	Model	Şasi no	Tür	Renk	Sahibi
BMW	740L	XZ68…	Sedan	Siyah	AA
VOLVO	S80	A32XT..	Sport	Sarı	BT
Ford	Taurus	X24R56	Hatchback	Bej	NB

Anahtar Alan

Te



Veri nesneleri arasındaki ilişkiler bire bir (1-1), bir den çoğa (1-N ya da N-1) ya da çoktan çoğa (M-N) tanımlanabilir. Örneğin:

1 - 1 ilişki : Bir İNSAN ancak bir ARABA sahibi olabilir.

1 - N ilişki: Bir İNSAN birden çok ARABA sahibi olabilir:

M - N ilişki: Birden çok İNSAN birden çok ARABA sahibi olabilir.

Veri Sözlüğü

- Nesne İlişki şemalarında belirtilen nesne özelliklerinin ayrıntılı tanımları Veri Sözlüğünde yer alır.

- Söz konusu ayrıntılı tanımlar genel olarak:
  - Veri Adı
  - Veri Eş-adı (Aynı veri için kullanılan diğer ad)
  - Nerede/nasıl kullanıldığı
  - İçerik tanımı

türünde bilgileri içerir.

Örnek: Kişi telefon bilgisinin tanımlanması

telefon no = [yer kodu | numara]
yer kodu = [212 | 242 | ..... | 312]
numara = * yedi basamaklı herhangi bir sayı *

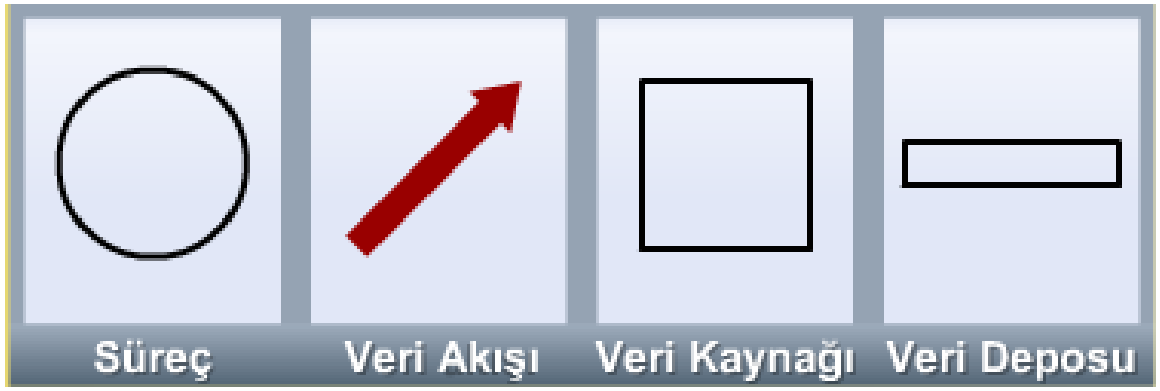
Veri Yapısı	Gösterim	Açıklama
	=	.. den oluşmuştur
Sıra Tanımlama	+	ve
Seçim	[|]	ya da
Yineleme	{ }^{n}	n kez yinelenir
	( )	seçenek veri
	* *	açıklama

Süreç/İşlem Modelleme Yöntemleri

- Süreç/İşlem modelleme yöntemleri, geliştirilecek sistemin süreç ya da işlemlerini ve bu süreçler arasındaki ilişkileri tanımlamak amacıyla kullanılan yöntemlerdir. En yaygın olarak kullanılan Süreç Modelleme Yöntemlerine örnek olarak:

Veri Akış Diyagramları

- VAD kullanılarak, geliştirilecek sistemin mantıksal modeli, 'Yukarıdan Aşağıya' bir yaklaşımla oluşturulur.
- Sistem önce en genel biçimiyle ele alınır, yalnızca dışsal ilişkileri incelenir. Daha sonra, sistemin iç yapısındaki süreçler ve bu süreçler arasındaki ilişkiler belirlenen bir ayrıntı düzeyine kadar modellenir.



VAD kümesi

- Temel olarak bir sistemin mantıksal modelinin süreç yapısı, üç tür Veri Akış Diyagramı çizilerek elde edilir.

Kapsam Diyagramı

Geliştirilecek bilgi sisteminin dışsal ilişkilerini göstermek amacıyla kulanılır. Yalnızca bir çember ve gerekli sayıda veri kaynağı ile aralarındaki veri akışlarının belirtiminden oluşur.

Genel Bakış Diyagramı

Bir Bilgi sisteminin ana işlevlerini, bu işlevlere ilişkin veri kaynaklarını, veri depolarını ve işlemler arasındaki ilişkileri içeren VAD, Genel Bakış Diyagramı olarak tanımlanmaktadır. Genel Bakış Diyagramı, kapsam diyagramının ayrıştırılmış ya da "patlatılmış" biçimidir. Bir bilgi sistemi için, bilgi sisteminin büyüklüğüne göre en çok bir kaç Genel Bakış Diyagramı çizilir.

Ayrıntı Diyagramı

Genel Bakış Diyagramında belirtilen her ana işlev, belirli bir ayrıntı düzeyine kadar (en fazla 7+/- 2) detay VAD oluşturularak ayrıştırılır. Genellikle detay VAD'larda, Veri kaynakları gösterilmez.

Tek

VAD Neyi Gösterir?

1. VAD, bilgi sisteminin durağan yapısını gösterir.

2. VAD, bilgi sisteminin süreçlerini, bu süreçler arasındaki veri akış ilişkilerini gösterir.

3. VAD, bilgi sistemi süreçleri ile ilgili olan kurum birimlerini ya da dış birimleri bilgi kaynakları olarak gösterir.

4. VAD, bilgi sistemi için gerekli olan ana veri depolarının neler olduğunu ve hangi süreçler tarafından kullanıldığını gösterir.

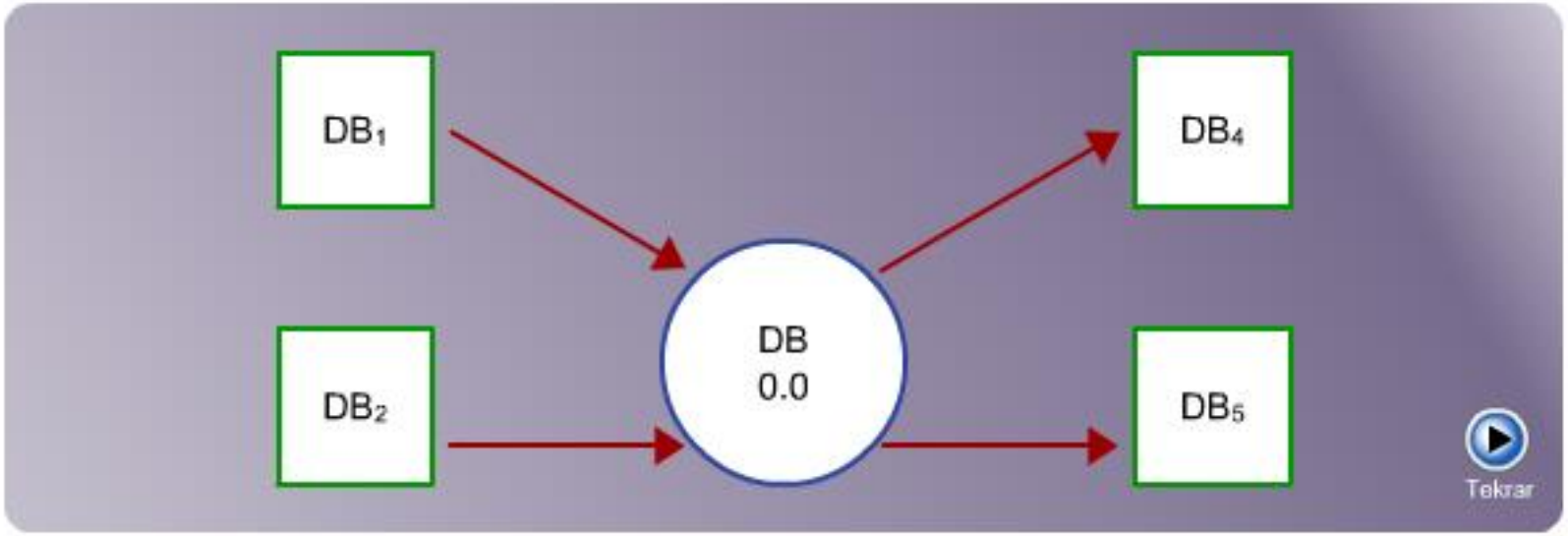
5. VAD, bilgi sistemi süreçlerini, yukarıdan aşağıya doğru ayrıştırarak gösterir. Böylelikle süreçler ve aralarındaki ilişki, en soyut (genel) düzeyden en ayrıntılı düzeye kadar belirli bir sıra düzen içerisinde belirtir.

VAD Neyi Göstemez?

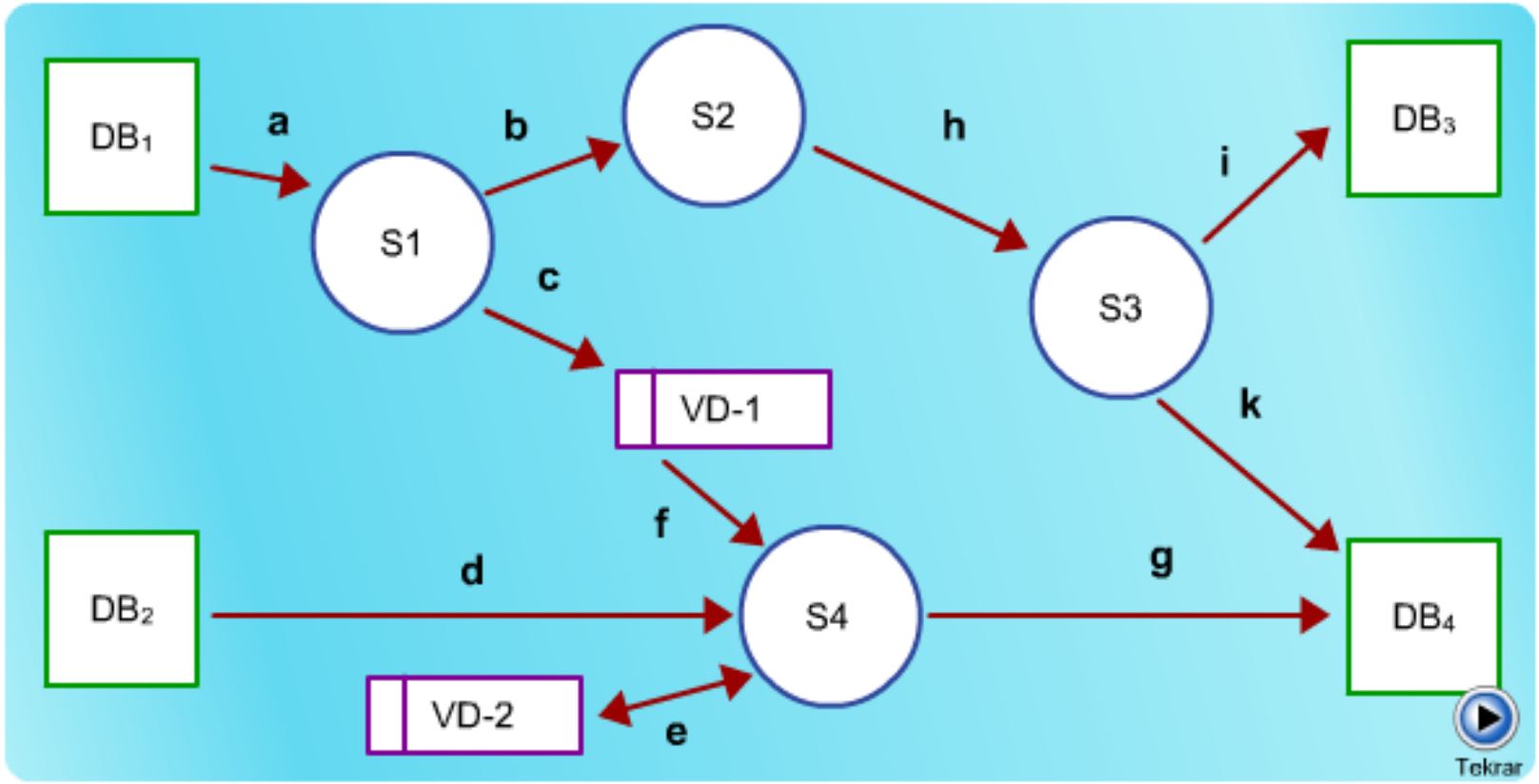
1. VAD bilgi sistemi süreçlerinin zamana ilişkin durumunu ve bu durumla ilgili bilgileri göstermez.

2. VAD, bilgi sistemi süreçlerinin kendi aralarındaki "karar" ilişkisini göstermez.

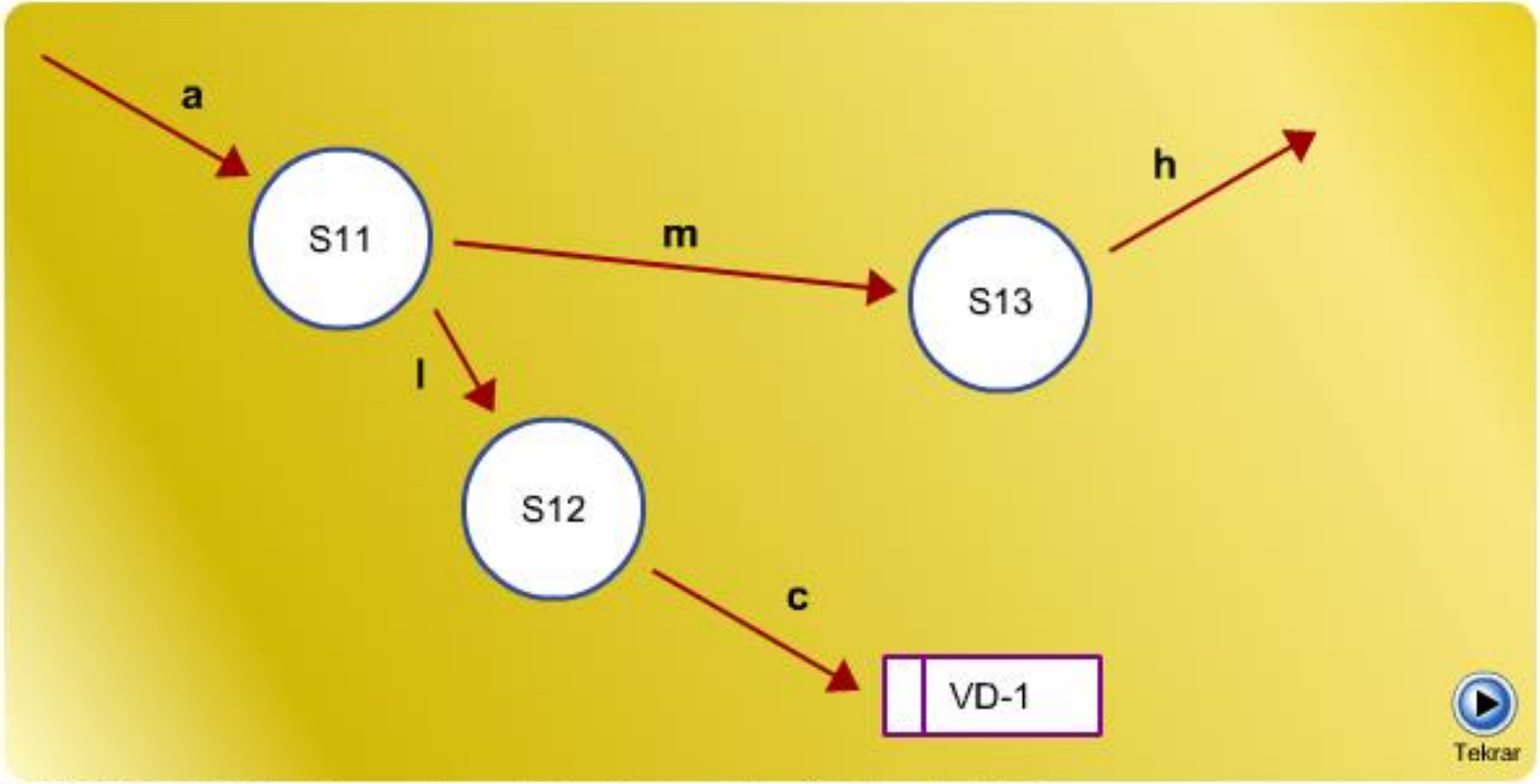
3. VAD, gerek bilgi sistemi süreçleri, gerek veri akışları gerekse bilgi kaynakları ve bilgi depoları için ayrıntı içermez.



Şekil 4.10a: Veri Akış Diyagramları - Kapsam Diyagramı Örneği
(DB: Dış birim, BS: Bilgi Sistemi)



Şekil 4.10b: Veri Akış Diyagramları - Genel Bakış Diyagramı Örneği
(S: Süreç, VD: Veri Deposu, a,b,c,d,e,f,g,h,i,k : Veri akışları)



Şekil 4.10c: Veri Akış Diyagramları - Detay Diyagram Örneği (Süreç-1 detayı)
(a,b,c,l,m : Veri akışları)

Süreç Tanımlama Dili (STD)

- Veri akış diyagramlarında isimleri belirtilen, aralarındaki ilişkiler gösterilen ve yukarıdan aşağıya ayrıştırılmış olan bilgi sistemi süreçlerinin iç yapılarını belirtmek amacıyla kullanılan araç, yöntem ya da gösterim biçimleri Süreç tanımlama dili olarak tanımlanmaktadır.

- Düz Metin,
- Şablon,
- Yapısal İngilizce

Düz Metin

Üçgeni İncele süreci üçgenin kenar boyutlarını (A,B,C) girdi olarak alır. Süreç önce bütün bu değerlerin pozitif olup olmadığını denetler. Eğer değerlerden biri negatif ise hata iletisi verir. Süreç, tüm kenar boyutlarının bir üçgeni belirleyecek biçimde geçerli olup olmadığını denetler. Eğer geçerli değerler ise üçgenin türünü (eşkenar üçgen, ikizkenar üçgen, çeşitkenar üçgen) belirler. Değil ise hata iletisi verir.

Şablon

Süreç: Üçgeni İncele
Girdi: Üçgenin kenar boyutları (A,B,C)
Çıktı: Üçgen Türü, Hata iletisi
İşlem: A,B,C değerlerin pozitif olup olmadığını denetle. Eğer değerlerden biri negatif ise hata iletisi ver. A,B,C değerlerinin geçerli olup olmadığını denetle. Eğer geçerli değerler ise üçgenin türünü (eşkenar ücgen, ikizkenar üçgen, çesitkenar üçgen) belirle. Değil ise hata iletisi ver.

Yapısal İngilizce

Procedure: Üçgeni İncele

Üçgenin kenar boyutlarını oku;
if herhangibir boyut negatif **then** hata iletisi ver; **endif;**
if en büyük kenar diğer ikisinin toplamından küçük **then**
  begin
  eşit kenar sayısını belirle;
  if üç kenar eşit then tür eşkenardır; **endif;**
  if iki kenar eşit then tür ikizkenardır; **endif;**
  if hiç bir kenar eşit değil then tür çesitkenardır; **endif;**
  üçgen türünü yaz;
end;
else değerler üçgeni belirtmiyor iletisi ver; **endif;**
endproc

Tekrar

KULLANICI ARAYÜZ PROTOTİPLEME (KAP)

- Gereksinim tanımlama ya da sistem çözümleme çalışmasının önemli bir bileşeni, oluşturulacak bilgi sistemine ilişkin girdi ve çıktı gereksinimlerinin tanımlanmasıdır.
- KAP Yöntemi, gereksinim çalışmasının hemen sonunda kullanıcıya gösterilecek bir prototip yazılım hazırlanmasını içermektedir. Söz konusu prototipin gerçekten içsel olarak çalışmayan ancak ekranlar, menüler ve bunların aralarındaki geçişlerin çalıştığı bir yazılımdır.

KAP İçin Ekranlar Nasıl Hazırlanır?

K

1. KAP hazırlama süresi uygulamanın boyutu ne olursa olsun, ekranlar ve ekranlar arası geçişlerin hazırlanması için gerekecek iş gücü ve zaman, sistem çözümleme için ayrılan zamanın %5'ini aşmamalıdır.

A

2. Bir özellik yalnızca bir kez gösterilmelidir. Bir özellik birden fazla ekranda yer alsa bile yalnızca bir ekranda gösterilmelidir. Örneğin, şehir listesini içeren bir kombo listesi, bir kaç şehir bilgisi içermeli ve yalnızca bir ekranda kulanıcıya gösterilmelidir. Başka ekranlarda aynı liste varsa, gösterilen ile aynı olduğu belirtilmelidir.

P

3. KAP hiçbir içsel işlem içermemelidir. Örneğin: bir ekranda kişilerle ilgili sorgulama yapıldığını düşünelim. Adı soyadı bilinen bir kişinin, bu bilgi verildiğinde, diğer ayrıntı bilgilerinin ekranda görüntüleneceğini varsayalım. Kullanıcı, Ad Soyad bilgisini girecek ve ilgili sorgulama tuşunu etkinleştirdikten sonra ekranda hep aynı bilgiler görünecektir. Yani, KAP, Ad soyad bilgisine göre KİŞİ tablosundan arama yapmayacak, program içerisinde hep aynı sonucları ekrana verecektir. Çünkü Kişi tablosunun yaratılması ya da kullanılması söz konusu değildir. Burada amaç, kullanıcıya, sorgulama biçimini ve sorgulama sonucunda elde edilecek bilgileri açıklamaktır. Sorgulanan bilginin içeriği bu aşamada önemli değildir. Kulanıcıdan, sorgulama sonucu gelen bilgi türlerine ilişkin, sorgulama işleminin işlem akışına ilişkin bilgi alınması hedeflenir.

KAP İçin Raporlar Nasıl Hazırlanır?

- Bilgi sisteminden yazıcı çıktısı biçiminde alınması istenen raporlar, bir metin düzenleyici (örneğin MS Word) aracılığı ile hazırlanır ve belirli bir biçimde numaralandırılır.

ÖRNEK

Geliştirilecek bilgi sisteminin İnsan Kaynakları Yönetim Bilgi Sistemi (IKYS) olduğunu varsayalım. IKYS Atama Modülüne ilişkin raporların örneklerini MS Word ile hazırlayarak bu raporlara IKYS-AT-R1, IKYS-AT-R2... vb. numaralar verelim. Kullanıcı için hazırladığımız prototipi, kullanıcıya gösterirken, kullanıcı rapor almak amacıyla ilgili tuşa bastığında ya da fare imleci ile ilgili rapor alma düğmesini etkinleştirildiğinde KAP "Şu anda yazıcıdan IKYS-AT-R1 no'lu rapor dökülmekte" iletisi türünde bir ileti verecektir. Kullanıcı bu iletiyi gördüğünde, daha önceden MS Word ile hazırlanmış olunan ve IKYS-AT-R1 biçiminde numaralanmış raporu kullanıcı ya göstererek, ileride yazılımı bitirip, sistemi hazır hale getirdiğimizde bu rapor un aynısının yazıcıdan alınacağını belirtiriz. Kulanıcı bu durumda bize, rapor alanları, raporun tetiklendiği ekranla ilgili geri bildirimlerini ve değişiklik isteğini iletir. Bu değisiklikler yapıldıktan sonra söz konusu raporla ilgili gereksinimler kesinleştirilmiş olur.

ÇÖZÜMLEME ÇALIŞMASI NASIL DEĞERLENDİRİLİR?

- Sistem çözümleme çalışması sonuçlandıktan sonra, elde edilen ara ürünün (mantıksal model) istenenleri karşılayıp karşılamadığının belirlenmesi amacıyla değerlendirilmesi gerekir. Bu nedenle temel olarak:

Kullanıcıların benimsemesi ve onayı

Tutarlılık, tamlık ve uygunluk

Tasarım çalışması için yeterliliği

Tamlık ve Tutarlılık

Tamlık, bilgi sitemi ya da yazılımın tüm öğeleri ve bunların arasındaki ilişkilerin tanımlanmasını gerektirir. Tutarlılık ise, önerilen modelin kendi içerisinde hatasız, çelişkisiz olması anlamındadır. Tamlık ve tutarlılık denetimi, bir programlama dilinde yazılmış bir programın söz dizim kurallarının denetimine benzer. Örneğin, modelleme aracı olarak VAD kullanıldığında;

- Bütün süreçlerin girdi ve çıktıları var mı?
- Süreçler, veri akışları, dış birim ve veri depoları uygun şekilde adlandırılıp numaralandırılmış mı?
- VAD' de ki tüm isimler veri sözlüğünde yer almış mı?
- Planlama aşamasında öngörülen belirtimlerin hepsi ele alınmış mı?

Olurluluk

- Olurluluk, sistem çözümleme sırasında yapılan çalışmanın, planlama aşamasında yapılan çalışmaya uygunluğunun belirlenmesi için yapılan çalışmaları içerir.
- Maliyet kestirim çalışması yinelenir, sapmalar saptanır, kaynaklar yeniden planlanır. Sapmalar olduça fazla olursa, yapılan çalışmanın yeniden gözden geçirilmesi gerekir.
- Bu aşamada, maliyet kestirimi için daha fazla bilgi mevcuttur. VAD yöntemi, maliyet kestirim yöntemlerinin kolayca uygulanabilmesini olanaklı kılar. Örneğin İşlev Noktalarının belirlenmesinde VAD'den:

VAD'den elde edilenler

Girdi ve Çıktı Sayıları: Girdi ve Çıktı sayıları doğrudan genel bakış diyagramındaki veri akışlarının incelenmesiyle elde edilir.

Kütük Sayısı: Genel bakış diyagramlarındaki veri depoları sayılarak elde edilir.

Sorgu Sayısı: Sorgu sayısı genel bakış diyagramı ve detay diyagramlardaki süreçlerin incelenmesiyle elde edilir.

Arayüz Sayısı: Arayüz sayısı, kapsam diyagramındaki dış birimlerle olan ilişkiler incelenerek elde edilir.

Alınan Dersler

- Sistem Çözümleme aşaması, uç kullanıcı ile iletişimin en fazla olduğu aşamadır. Kullanıcıların Bilişim teknolojileri konusundaki bilgi düzeyleri genelde yok denecek ölçüde düşüktür. Bu nedenle kullanıcı ile olan iletişimin olabildiğince kullanıcıya görsel olanaklar sunularak yapılması önemlidir. Bu nedenle, özellikle kullanıcı arayüzünün belirlenmesi için en etkili yöntem olarak Kullanıcı Arayüz Prototipleme yöntemi önerilmektedir.

- KAP yöntemini kullanıcılarla birlikte tartışırken, "iş senaryoları" oluşturmanın oldukça yararı vardır. Her iş senaryosu, bir iş probleminin çözümüne karşılık gelecek biçimde hazırlanmalı ve KAP üzerinde sınanmalıdır. BU yolla, kullanıcı, işeride bilgisayarlı uygulamaya geçildiğinde, nasıl bir ortamla karşı karşıya geleceğine ilişkin fikir sahibi olur.

Bu yöntemin kullanılması ileride ortaya çıkabilecek belirsizlik ve riskleri de büyük ölçüde ortadan kaldırır. Üretim sonucunda elde edilecek ekran ve rapor görüntülerinin bu aşamada olabildiğince kesinleştirilmesi, hem üretim yapan yazılım mühendislerinin hem de kullanıcının işini oldukça kolaylaştırır.

Alınan Dersler

Kullanıcılar, genellikle görüşme tutanaklarını imzalamakta çekingenlik gösterirler.'Siz hazırlayın, bize gönderin bizde imzalar size göndeririz ' biçiminde yöntemler önerirler.

Görüşme tutanaklarınızı anında iki kopya olarak tutun ve hemen görüşme bitiminde imzalatma yöntemini benimseyin ve uygulayın. Aksi durumda, ileride ortaya çıkabilecek sorunlarda kaybeden taraf siz olursunuz.

Alınan Dersler

Tüm görüşme ve toplantı kayıtlarınızı olabildiğince bilgisayarlı ortamda saklayın. Aradan, uzun bir süre geçtikten sonra, eski kayıtlara erişimde büyük kolaylıklar sağladığını göreceksiniz.

Bu amaçla Visual Source Safe türü basit bir araç bile kullanabilirsiniz.

Alınan Dersler

VAD yöntemi, kullanıcı tarafından algılanma açısından oldukça etkili bir yöntemdir. Bu durum, değişik projelerde görülmüştür.

Kullanıcıya, görüşme ile ilgili olarak düzeylendirilmemiş bir VAD hazırlayıp gönderdiğinizde, kullanıcının bu VAD'nı kendi başına yeniden çizebildiği gözlemlenmiştir.

Alınan Dersler

CASE aracı kullanımının modelleme çalışmasını oldukça hızalandırdığı gözlemlenmiştir. Niteliği ne olursa olsun mutlaka bir CASE aracı kullanın.

CASE aracı kullanımı ayrıca, daha sonraki aşamalarda oluşabilecek ve mantıksal modele yansıtılması gereken günlemelerin kolayca yapılmasını sağlar.

Sorular

1. Sistem Çözümleme çalışmasının amaç ve önemini belirtiniz.

2. Gereksinim modelleme çalışmasında neden grafiksel araç ve yöntemler daha sıklıkla kullanılır?
3. Mevcut sistemin incelenmesi için kullanılabilecek yöntemleri açıklayınız.
4. İş senaryosunu tanımlayınız. Bir PERSONEL bilgi sistemi uygulaması için üç iş senaryosu örneği veriniz.
5. Çevrenizde var olan bir CASE aracını inceleyip, sistem çözümleme çalışması ile ilgili olarak hangi olanakları içerdiğini araştırınız.
6. Gereksinimlerin belirlenmesi sırasında ortaya çıkabilecek riskleri sıralayınız.

Sorular

7. Beş adet açık uçlu soru örneği veriniz.
8. Beş adet kapalı uçlu soru örneği veriniz.
9. İstediğiniz bir konuda 20 soruyu içeren bir anket hazırlayınız. Anket yanıtlama problemlerini dikkate alınız.
10. Kaset, CD satan ve kiralayan bir müzik dükkanında yapılan işlemleri, yapısal VAD kullanarak modelleyiniz. Süreç tanımlama dili olarak düz metin kullanınız. Veri yapısını Nesne-İlişki diyagramları ile modelleyiniz.

11. 8. Soruda oluşturduğunuz VAD üzerinde

- Dönüştürme akış özelliklerini,
- Ara işlem akış özelliklerini

gösteriniz.

Sorular

12. Kullanıcı arayüz prototiplemenin amacı nedir? Yararlı ve aksak yönleriyle belirtiniz.

13. VAD, süreç tanımlama dili ve nesne ilişki şemaları kullanılarak yapılan bir modellemenin, fiziksel tasarıma nasıl yardımcı olacağını açıklayınız.

14. Tasarımla çözümleme arasındaki ilişkiyi belirtiniz.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM

- Yazılım :
  - Herhangi bir boyuttaki herhangi bir tür donanımda çalışan bilgisayar programını VE,
  - Basılı veya elektronik ortamdaki her tür dokümanı içeren ürün.
    - Dokümanlar yazılım mühendislerine ve son kullanıcıya yönelik olabilir.
- Yazılım hem bir üründür, hem de başka ürünler elde etmeye yarayan bir araçtır.
- Yaşam döngüsü: Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen süreç.
- Yazılım fiziksel bir ürün olmadığı için aşınmaz, ancak zamanla yetersizleşebilir.
  - Değişim kaçınılmazdır: Yazılım, yaşam döngüsü süresince değişikliklere uğrar.
  - Değişiklikler, yazılımda yeni hatalar oluşturabilir.
  - Yeni hatalar tam olarak düzeltilmeden yeni değişiklikler gerekebilir.
- Çözüm: Yazılım mühendisliği ilkelerine uyularak daha iyi tasarlanmış yazılım.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM TÜRLERİ

- Sistem Yazılımı :
  - Diğer programlara hizmet sunmak üzere hazırlanmış programlar.
  - Derleyiciler, işletim sistemleri, vb.
  - Karmaşık olsa bile belirli bilgi yapıları ile uğraşır.
    - Belirli: Girdi, işlem ve çıktıların sırası ve zamanlamasının önceden tahmin edilebileceği sistemler.
- Mühendislik Yazılımı / Bilimsel Yazılım :
  - Mühendislik ve bilimsel hesaplamalarda kullanılmak üzere hazırlanmış programlar.
  - Büyük hacimli verilerle uğraşır.
  - “Numara öğütmek / Number crunching”.
- Gömülü (Embedded) Yazılım :
  - Bir ürün veya sistemin bir parçası olup, bu sistemin kendisi ve/veya son kullanıcısı için denetim işlemleri yürüten programlar.
  - Gerçek zamanlı uygulamalardır.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM TÜRLERİ

- Şirket Yazılımı :
  - Belirli ticari iş gereksinimlerine yönelik programlar.
  - İş süreçleri ile ilgili bilgiye sahip olmalıdır.
  - Genellikle müşteriye özel tasarlanır.
  - Veri dönüştürme ve değerlendirme uygulamaları, iş süreçlerinin kimi zaman gerçek zamanlı izlenilmesi, vb.

- Uygulama Yazılımı :
  - Product-line, shrink-wrapped, off-the-shelf, vb.
  - Farklı müşteriler tarafından kullanılabilecek genel amaçlı yazılımlar
  - Cari hesap uygulamaları, çeşitli otomasyon programları, kelime işlem uygulamaları, vb.

YAZILIM TÜRLERİ

- Ağ Yazılımları :
  - Ağ üzerinden haberleşerek hizmet almaya veya vermeye yönelik uygulamalar.
  - E-ticaret, B2B, B2C, web servisleri, web tarayıcıları, vb.
  - Şirket yazılımları ile etkileşimde bulunabilirler.
- Yapay Zeka Yazılımları :
  - Sayısal olmayan algoritmalarla karmaşık sorunları çözmeye yönelik yazılımlar.
  - Robotik, uzman sistemler, örüntü tanıma (ses ve görüntü), vb.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIMI ETKİLEYEN EĞİLİMLER

- Yaygınlaşan Bilgi-İşlem :
  - Hesaplama gücünün giderek küçülen alanlara sıkıştırılabilmesi, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Yaygınlaşan Haberleşme Ağı :
  - Kablosuz ağların yaygınlaşması, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Özgür / Açık Kaynak Yazılım :
  - Gevşek bir ekip tarafından geliştirilen yazılım, daha anlaşılır ve geliştirilebilir olmalıdır.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

ESKİ YAZILIM (Legacy Software):

- İş sürecinin önemli bir parçası olan ve çok uzun süredir kullanılan yazılımlar.
- Eski yazılımda bulunabilecek olumsuzluklar:
  - Eksik veya hatalı dokümantasyon
  - Zamanla karmaşıklaşmış kod
  - Esnek olmayan yapı
  - Eski donanımla çok sıkı ilişki
  - Yazılım mühendisliğindeki gelişmelerden yoksunluk nedeniyle düşük kalite.
- Eski yazılımın değiştirilmesini gerektiren nedenler :
  - İş alanındaki yeni gereksinimler
  - Güncel sistemlerle birlikte çalışabilmesi için uyumluluk kazandırılması
  - Donanımın ömrünün dolması nedeniyle daha güncel ortama taşınma gerekliliği

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM HAKKINDAKİ YANILGILAR: MÜŞTERİ AÇISINDAN

- Programın yazılmasına başlanması için amaçları genel olarak belirlemek yeter, ayrıntılar sonra kararlaştırılabilir. Nasıl olsa yazılım esnektir.
  - Belirsiz gereksinimler, çürük atılmış temele benzer.
- Yazılım esnektir. Değişen gereksinimler kolayca sisteme uyarlanabilir.
  - Yazılım yaşam döngüsünde ilerledikçe, değişen gereksinimleri yazılıma uyarlamanın bedeli üstel olarak artar.
- Sonuç: Yazılım esnek bir oyun hamurundan çok kil veya cam gibidir.
  - Çevik süreçlerle esnekliğin arttırılması hedeflenmektedir.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM HAKKINDAKİ YANILGILAR: PROGRAMCI AÇISINDAN

- Yazılımı tamamlayınca işimiz biter.
  - Yazılım üstünde harcanan çabanın yarısından fazlası, yazılımın müşteriye ilk teslimatından sonra harcanmaktadır.
- Yazılımı tamamlamadan kalitesini ölçemem.
  - Kalite güvence yöntemleri yazılım hayat döngüsünün her aşamasında uygulanabilir.
  - Çözümleme sürecinde dahi kullanılabilecek kalite ölçütleri bulunmaktadır.

- Yazılım ürünü, doğru çalışan program demektir.
  - Dokümantasyonu unutmayın!
- Yazılım mühendisliğinin gereklerini uygulayarak boşuna çaba harcıyoruz. Sayfalarca gereksiz doküman hazırlayacağımıza zamanımızı ve çabamızı daha verimli kullanalım.
  - Haritası olmayan yolunu kaybeder.
  - Kalite için harcanan çaba, karşılığını yazılım hayat döngüsünün ilerleyen aşamalarında fazlasıyla ödeyecektir.

YAZILIM HAKKINDAKİ YANILGILAR: İDARİ

- İşler yetişmiyorsa takıma yeni programcılar ekleriz.
  - Yazılım hayat döngüsü içerisinde ilerledikçe, yeni elemanların yazılıma hakim olması üstel olarak zorlaşır. İşler daha da gecikir.
- Elimizde yazılım geliştirme ile ilgili zengin bilgi kaynakları bulunmakta. İşimiz kolay olacak.
  - Kaynaklar tam, doğru ve güncel mi? Bu kaynaklar projemize uyarlanabilir mi? Ekip bu içeriği özümseyebilmiş mi?
- Geliştirmesini üstlendiğim yazılımı tamamen veya kısmen fason yaptırırım.
  - Proje ilerlemesini kendi içinde denetleyemeyen bir firma, dışarıya verdiği işi izlemekte de zorlanacaktır.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme (Analysis)
- Tasarım (Design)
- Gerçekleme (Implementation)
- Sınama (Testing)
- Bakım (Maintenance)

ÇÖZÜMLEME

- Çözümleme: Bir şeyi anlayabilmek için parçalarına ayırmak.
- Sistemi anlamaya yönelik çalışmalardan ve **üst düzey** planlama eylemlerinden oluşur.
  - Uygulama alanı
  - Kullanıcı gereksinimleri
  - Program parçaları arasındaki üst düzey etkileşimler (NYP'deki parçalar: sınıflar ve nesneler)
- “Bir sorunu anlamadan çözemezsiniz.”

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

TASARIM

- Tasarım: Bir araştırma sürecinin çeşitli dönemlerinde izlenecek yol ve işlemleri tasarlayan çerçeve.
- Çözümleme ile anlaşılan sorun tasarım aşamasında kağıt üzerinde (!) çözülür.
- Yazılım ↔ Tasarıma yönelik şemalar (NYP'de UML şemaları), elektronik ↔ devre şemaları, mimari ↔ kat planları

GERÇEKLEME

- Eldeki tasarım, bir programlama dili ile kodlanır.

YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

SINAMA

- Sınama neden önemlidir?
  - Yazılım sürecinde ilerledikçe, ortaya çıkabilecek hataların giderilme maliyeti üstel olarak artar.
  - Aksi gibi, hataların büyük çoğunluğu isteklerin belirlenmesi ve tasarım aşamalarında ortaya çıkar.
  - Bu yüzden: Erkenden, sık sık ve kolay sınama yapın.

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

BAKIM

- Yazılımın faaliyete geçirilmesinden sonra sistemde yapılan değişikliklerdir.
  - Yazılım hatalarının düzeltilmesi:
    - Kodlama hataları
    - Tasarım hataları (!)
    - Gereksinim ve analiz hataları (!!)
  - Sistemin işlevlerini değiştirme veya işlevlere eklemeler/çıkarmalar (!),
  - Yazılımın farklı bir ortama taşınması (yazılım, donanım, iklim),

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

BAKIM

- Yeniden mühendislik (Refactoring / Software re-engineering)
  - Teknik bakış açısı: Yazılımın işlevini değiştirmeden iç yapısını değiştirmek.
  - Olası eylemler:
    - Yazılımın belgelendirilmesi
    - Yazılımın güncel bir programlama diline geçirilmesi (!)
    - Tasarımın iyileştirilmesi/değiştirilmesi
    - Yazılımın farklı bir ortama taşınması

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

YAZILIM YAŞAM DÖNGÜSÜ

- Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen aşamalardır.
- Döngü: Kullanım dışı bırakılan yazılımın yerine yenisi hazırlanabilir.
- Döngünün aşamalarının belirlenmesi ve tanımlanması ile yazılım geliştirme modelleri/süreçleri elde edilir.

YAZILIM GELİŞTİRME SÜREÇLERİ/MODELLERİ

- Yazılım geliştirme bir süreçtir
  - Süreç: Önceden belirlenmiş adımlardan oluşan iş akışı.
  - Yazılım geliştirme modelleri, sürecin yapısını ve adımlarını belirler.
  - Önceden ve iyi planlanmış bir süreç, zamanında ve ‘kaliteli’ bir ‘ürün’ elde edilmesini sağlar.
- Çeşitli modellerin kendine özgü avantaj ve dezavantajları vardır.
  - Gerçeklenecek projeye uygun modelin seçilmesi gerekir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ŞELALE MODELİ

- Ardışıl Model / Şelale Modeli (Sequential / Waterfall)
  - Adımlar: Analiz (Çözümleme) – Tasarım – Kodlama – Sınama – Bakım.
  - Bir adımın tamamlanmasından sonra diğerine geçilir.
  - Eksiklikler veya hatalar farkedilirse bir önceki adıma geçilir.
- Artılar:
  - En eski model, yaygın kullanımda.
  - İyi tanımlanmış adımlar.
- Eksiler:
  - Son ürünün eldesi uzun süreceğinden müşteri sabırlı olmalıdır.
  - Adımları geride bıraktıkça, ilerleyen aşamalarda karşılaşılan hataların düzeltilmesi üstel olarak zorlaşmaktadır.
  - Bir çok ‘müşteri’ de gereksinimleri eksiksiz ve kesin belirtmekte zorlanmaktadır.
- Sonuç: Hiç model kullanmamaktan iyidir!

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÖN ÜRÜN MODELİ

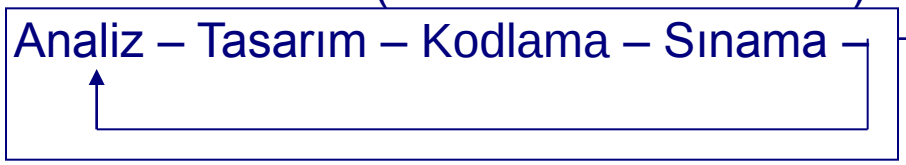
- Ön ürün modeli / Prototip modeli
  - Adımlar: Müşteriyi dinle – Ön ürün oluştur – Müşteri ön ürünü dener –

- Artılar :
  - Kullanıcı gereksinimlerinin daha iyi elde edilmesi.
  - Kullanıcının erkenden ürünü değerlendirmeye başlayabilmesi.
- Eksiler :
  - Ön ürün mükemmel değildir.
  - Eksik ürün zaman ve maliyet kısıtlamaları nedeniyle olgunlaşmadan canlı kullanıma alınabilmektedir.
- Sonuç: Prototip oluşturmayı başlı başına bir model olarak kullanmamalı, daha olgun bir modelin analiz aşamasında kullanılacak bir araç olarak ele almalı ve prototip ürünü silip atmalı.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ARTIMSAL / YİNELEMELİ MODELLER

- Artımsal / Yinelemeli Modeller (Incremental / Iterative)
  - Adımlar: Analiz – Tasarım – Kodlama – Sınama – → Bakım



  - Gereksinimler önemlerine ve birbirine bağımlılıklarına göre sıralanarak her yinelemede bunların bir kısmı tamamlanır.

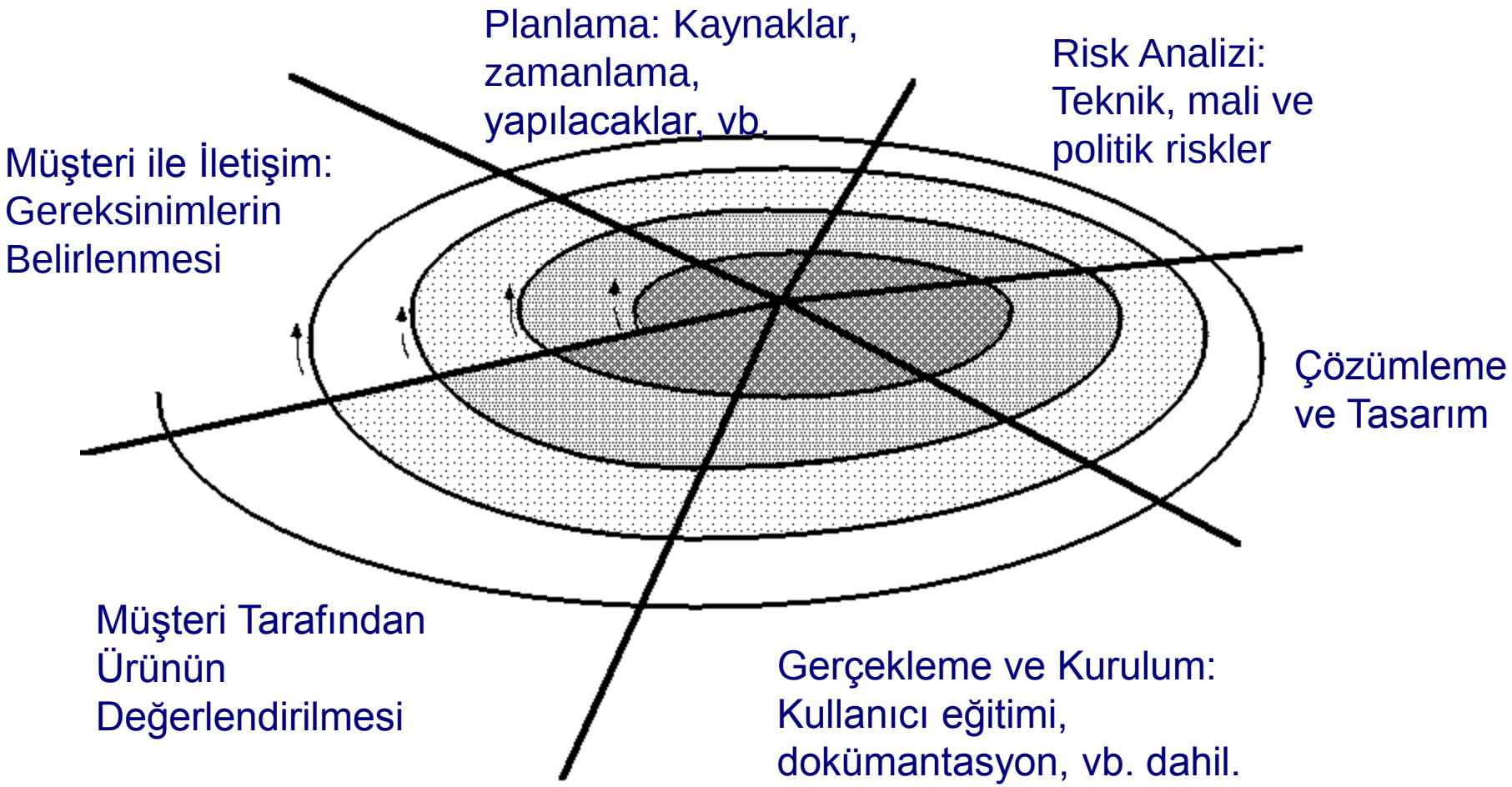
- Artılar :
  - Ön ürün modeli ve ardışıl modelin güçlü yönlerini kendinde toplayarak dezavantajlarını geride bırakmıştır.
  - Nesneye yönelik programlama metodolojisi ile uyum içerisindedir.

- Eksiler : Yazılımın küçük artımlarına fazla yoğunlaşmak, sistemin geneline bakıldığında kolayca görülebilecek sorunların gözden kaçmasına neden olabilir.

- Sonuçlar:
  - Sistemin genelini göz ardı etmemek şartıyla güçlü bir modeldir.
  - Güncel modellerin temelini yinelemeli ilkeler oluşturmaktadır.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

SARMAL (Spiral) MODEL



YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

Kazan-Kazan Sarmal Modeli (WINWIN Spiral Model)

- Paydaş: Yazılımın başarısı ve başarısızlığının etkileyeceği kişi ve kurumlar.
- Eksiler : Her planlama aşamasında bütçenin yeniden değerlendirilmesi, sabit bütçe isteyen(!) yöneticileri mutlu etmez.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

HIZLI UYGULAMA GELİŞTİRME (RAD: Rapid Application Development)

- Kısa geliştirme çevrimleri üzerinde duran artımsal bir model.
- Gereksinimler:
  - Uygulamanın yaklaşık/ortalama 3 aylık bölümlere ayrılabilmesi,
  - Yeterli sayıda bölümün eşzamanlı ilerlemesinin sağlanabilmesi,
  - Yazılımın bileşenlerden kurulabilmesi.
- Artılar:
  - Bu sürece uygun yazılım projelerinde verimliliğin artması.
- Eksiler:
  - Büyük ölçekli çalışmalarda yeterli sayıda bölümü eşzamanlı ilerletebilecek sayıda çalışanın bulunamaması.
  - Çalışanlar hıza uyum sağlayabilmelidirler.
  - Yüksek teknik risklere uygun değil.
- Sonuç:
  - Prototip geliştirmede kullanılması veya ana fikirlerinin diğer süreçlere uygulanması yerinde olacaktır.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

BİLEŞEN TABANLI (Component Based) UYGULAMA GELİŞTİRME

- Uygulamanın hazır yazılım bileşenlerinden oluşturulmasını öngörür.
- Aşamaları:
  - Konu alanı mühendisliği (Domain Engineering)
  - Aday bileşenlerin sınıflandırılması ve seçilmesi (Qualification)
  - Seçilen bileşenlerin kendi yazılımımıza uyarlanması (Adaptation)
  - Bileşenlerin bir araya getirilmesi (Composition)
- Artılar:
  - Yeniden kullanımın özendirilmesi (azalan giderler)
- Eksiler:
  - Uygun bileşenlerin bulunması gerekliliği
  - Bileşenlerin uyarlanması gerekliliği
- Sonuçlar:
  - Özellikle hızlı uygulama geliştirme olmak üzere, ana fikirleri çeşitli süreçlere uygulanabilir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Değişen gereksinimler, teknik riskler gibi önceden belirlenemeyen durumlara ve yazılım ürününü etkileyebilecek her tür değişikliğe karşı esneklik sağlayan süreçlerdir.

- Bireyler ve etkileşimler
- Çalışan yazılım
- Müşterinin sürece katılımı
- Değişikliklere uyum sağlamak

- Süreçler ve gereçler
- Ayrıntılı belgeler
- Sözleşme pazarlığı
- Bir planı izlemek

- Çevik süreçler, sağ taraftaki maddelerin yararını kabul etmekle birlikte, sol taraftaki maddelere daha çok önem vermektedir.
- Bir ilerleme olmaksızın yalnızca sürekli uyum sağlamak başarı değildir.
  - Yazılımın artımsal gelişimi
  - Müşteriye erken ve sık ürün teslimi
  - Başarımın birincil ölçütü çalışan yazılımdır.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Çevik süreci yürütecek ekibin özellikleri:
  - Yüz yüze görüşme, en etkili bilgi aktarım yoludur.
  - Takım üyeleri çevik yaklaşım hakkında eğitilmelidir.
  - Ekip üyelerinin ortak amacı, çalışan yazılım üreterek müşteriye zamanında teslim etmek olmalıdır.
  - Ekip üyeleri birbirleriyle ve müşteriyle işbirliği içinde olmalıdır.
  - Ekip üyeleri karşılıklı saygı ve güven içerisinde olmalıdır.
  - Ekipler hem teknik, hem de tüm proje hakkında kararlar verebilmelidir.
  - Boşuna harcanan çaba yoktur: Çözülen bir sorun gereksizleşse bile, çözüm sürecinde edilen deneyim ekibe ileri aşamalarda yararlı olabilir.
  - Kendi kendini düzenleme:
    - Ekibin kendisini yapılacak işe göre uyarlaması,
    - Ekibin kullanacağı süreci yerel ortama uyarlaması,
    - Üstünde çalışılan artımsal yazılım parçasını teslim etmek için gerekli çalışma zamanlamasını ekibin kendisinin belirlemesi.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Çevik Süreç Örnekleri:
  - Aşırı Programlama (XP: Extreme Programming)
  - Sürü (Scrum)
  - Çevik Modelleme

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama Artımsal Ürün

- Planlama:
  - Müşteri, kullanıcı öyküleri (KÖ) oluşturur.
    - Müşteri, öyküleri önemine göre derecelendirir.
    - Yaklaşık 3 haftada gerçeklenemeyecek öyküler varsa, ekip müşteriden bunları alt öykülere bölmesini ister.
  - Ekip ve kullanıcı, öykülerin sıradaki artımsal ürüne nasıl ekleneceğine karar verir:
    - Ya önce yüksek riskli öyküler gerçeklenir,
    - Ya da önce yüksek öncelikli öyküler gerçeklenir.
    - Her olasılıkta tüm öyküler kısa sürede (birkaç hafta) gerçeklenmelidir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama Artımsal Ürün

- Planlama (Devam):
  - İlk artımsal ürün projenin hızını ölçme amacıyla değerlendirilir:
    - Eldeki artımın hızına göre sonraki artımların teslim tarihleri belirlenir.
    - Aşırı sözler verildiği ortaya çıkarsa artımsal ürünlerin içeriği de yeniden kararlaştırılabilir.
  - Süreç ilerledikçe müşteri yeni öyküler ekleyebilir, eski öykülerin önceliğini değiştirebilir, öyküleri farklı şekillerde bölüp birleştirebilir, bazı öykülerden vazgeçebilir.
    - Bu durumda ekip kalan artımları ve iş planlarını uygun biçimde değiştirir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama Artımsal Ürün

- Tasarım:
  - Basit tasarım karmaşık gösterimden üstündür. (KISS: Keep It Simple, Stupid!)
  - CRC (Class-Resposibility-Collaboration) kartları ile yazılımın sınıf düzeyinde incelenmesi.
  - Karmaşık bir tasarımdan kaçınılamazsa işlevsel bir ön gerçekleme yapılır (Spike solution).
  - Refactoring teşvik edilir.
  - Bu aşamanın ürünleri CRC kartları ve ön gerçeklemelerdir (başka ürün yok).

ÇEVİK (Agile) SÜREÇLER

- Örnek CRC kartı:

Sınıf adları

Sınıf: Satış	
Kasada yapılan ödemeyi simgeleyen sınıf.	
Üst Sınıf(lar): Yok	
Alt Sınıf(lar): Yok	
Sorumluluk:	**İşbirlikçi:**
Satışın yapıldığı tarih ve saati saklamak	
Yapılan ödeme tutarını saklamak	Ödeme
Satılan malların listesine erişim	Mal

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama Artımsal Ürün

- Kodlama:
  - Önce birim sınamaları hazırlanır.
    - Programcı tarafından yapılan, sınıfların (NYP'de; yapısal'da fonksiyonlar, vb.'lerin) temel işlevselliklerini sınama amaçlı kod.
    - Sadece sınavı geçmeye yarayan kod yazılır (KISS).
  - Çift kişi ile kodlama:
    - Bir programcı eldeki sorunu çözerken diğeri çözümün genel tasarıma uygunluğunu gözetir ve kodlamanın takımın karar verdiği ölçütlere (kalite, vb.) uygunluğunu denetler.

- Sınama:
  - Birim sınamalarının otomatik çalıştırılması.
  - Müşterinin artımsal ürünü denemesi.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Sürü (Scrum)
  - Adımlar: Görev Listesi – Koşu – İşlev Gösterimi

- Görev Listesi:
  - Müşterinin belirlediği, kendisine değer sağlayan gereksinimler ve özellikler (kullanıcı öykülerinin bir başka tanımı).
  - Önceliklendirilmiştir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Sürü (Scrum)
  - Adımlar: Görev Listesi – Koşu – İşlev Gösterimi

- Koşu:
  - Görev listesinin maddelerinden biri seçilir ve önceden belirlenmiş kısa bir süre içerisinde (30 gün) gerçeklenir.
  - Koşu süresince ekibin her gün yaptığı toplantılar:
    - Kısa sürer (15dk).
    - Sürü önderi yönetir.
    - Herkesin sorduğu ve cevapladığı üç ana soru:
      - Son toplantıdan bu yana ne yaptınız?
      - Karşılaştığınız engeller nelerdir?
      - Yarınki toplantıda neleri başarmayı hedefliyorsunuz?

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Sürü (Scrum)
  - Adımlar: Görev Listesi – Koşu – İşlev Gösterimi

- İşlev Gösterimi:
  - Müşterinin yeni işlevi sınaması.
  - Gösterime tüm işlevler eklenilmeyebilir.
    - Gerçeklenen işlevlerin gösterimi.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

ÇEVİK (Agile) SÜREÇLER

- Çevik Modelleme
  - Bir amaç için modelleme yapın:
    - Neyi, kime, hangi düzeyde anlatmak istiyorsunuz?
    - Buna göre uygun modelin ve ayrıntılandırmanın seçimi .
  - İçerik sunumdan daha önemlidir.
    - Gerekli bilgiyi içermeyen hatasız model işe yaramaz!
  - Kullandığınız modelleme yolunun özünü ve modellerinizi oluşturmak için kullanacağınız araçları iyi öğrenin.
- DİKKAT: Önemli olan dengeyi korumaktır.
  - Çevik çalışacağız diye serseri programcı olmayın.
  - Disiplinli çalışacağız diye sırtınızda tuğla çuvalı taşımayın.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

NESNEYE YÖNELİK ÇÖZÜMLEMENİN TEMELLERİ

- Çözümleme: Bir şeyi anlayabilmek için parçalarına ayırmak.
- Sistemi anlamaya yönelik çalışmalardan ve **üst düzey** planlama eylemlerinden oluşur.
  - Uygulama/problem alanının anlaşılması.
  - Kullanıcı gereksinimlerinin anlaşılması.
  - Koddaki sınıflar ve nesneler ile bunların arasındaki üst düzey etkileşimlerin belirlenmesi: Çözümleme modelinin oluşturulması.
- “Bir sorunu anlamadan çözemezsiniz.”

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

UYGULAMA ALANININ ÇÖZÜMLENMESİ (DOMAIN ANALYSIS)

- Amaç, uygulama alanını anlamak ve elde edilen bilgileri analiz modeline taşımaktır.
- Uygulama alanı hakkında bilgi edinilebilecek kaynaklar:
  - Teknik literatür
  - Mevcut uygulamalar
  - Müşteri anketleri
  - Uzman tavsiyeleri
  - Mevcut ve gelecekteki gereksinimler
- Problem alanı hakkında bilgi edinmeden “müşterinin dilinden konuşamazsınız”.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

GEREKSİNİMLERİN BELİRLENMESİ

- Gereksinimler belgesi:
  - Müşterinin programdan beklentilerini anlatan, doğal konuşma dili ile yazılmış belge.
- Örnek gereksinimler belgesi:

NextGenPOS Perakende Satış Programı

Eski yazılım ihtiyaçlarımızı karşılayamadığından, yenilenecek donanımla birlikte perakende satış programımızın da yenilenmesine gerek duyuyoruz. Program kasada yapılan alış-veriş işlemlerine yardımcı olmalıdır. Yapılan her işlem program tarafından saklanmalı; mali bilgiler harici bütçe sistemine, mal çıkış bilgileri ise harici envanter sistemine iletilmelidir. Saklanan işlemler üzerinde daha sonra raporlamalar ve analizler yapılabilmelidir. Sistem yapılan alış-verişler karşılığında müşteriye fiş vermelidir. Yapılan her satış için KDV de hesaplanarak ayrıca belirtilmelidir. Şirketimizin birden fazla şubesi olup tüm şubelerdeki işlemler merkezi sunucuya iletilmelidir.

- Doğal dille yazılmış gereksinimler belgesinden kullanım senaryolarına geçiş yapılır.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

GEREKSİNİMLERİN BELİRLENMESİ

- Kullanım senaryoları:
  - Programın yapacağı işleri ayrıntılı adımlarla ve belli kurallara uyarak anlatan belgeler.
- Kullanım senaryolarının oluşturulmasındaki amaç:
  - Ürünün sağlaması beklenen işlevleri ve ürünün çalışma ortamını belirlemek,
  - Son kullanıcı ve yazılım ekibi arasında bir anlaşma zemini belirlemek,
  - Son kullanıcı ve sistemin birbirleri ile nasıl etkileşimde bulunacağını açık ve belirsizlikten uzak olarak tanımlamak,
  - Doğrulama testleri için bir zemin oluşturmak.
- Bir kullanım senaryosunun bölümleri:
  - Giriş bölümü: Sistemin neyi hangi koşullar ve sınırlar içerisinde yapması gerektiğini anlatır.
  - Ana senaryo / Ana başarılı akış: Her şeyin yolunda gitmesi halinde yürütülecek eylemler.
  - Alternatif senaryolar: Bir aksilik olması halinde yapılacak işlemler.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

KULLANIM SENARYOSU: Satış İşlemi

Birincil Aktör: Kasiyer.

İlgililer ve İlgi Alanları:

- Kasiyer: Doğru ve hızlı giriş ister, kasa açığı maaşından kesildiğinden ödeme hataları istemez
- Satıcı: Satış komisyonlarının güncellenmesini ister
- Müşteri: En az çaba ile hızlı hizmet ister. Vergi iadesinde veya mal iadesinde kullanmak üzere fiş ister.
- …

Ön Koşullar:

- Kasiyerin kimliği doğrulanır.

Son Koşullar:

- Satış kaydedilir. KDV doğru olarak hesaplanır. Mali kayıtlar ve envanter kayıtları güncellenir. Komisyonlar kayıt edilir. Fiş yazılır. Ödeme emri onayları kaydedilir.

- Dikkat: Kullanım senaryosunda yer alacak her şey, verilen ilgi alanlarına giren şeyler olmalıdır.
- Aktör: Sistem ile etkileşimde bulunan varlıklar.
  - İnsan
  - Yazılım veya donanım.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

KULLANIM SENARYOSU: Satış İşlemi

Ana Senaryo:

1. Müşteri kasaya alacağı mallarla gelir.
2. Kasiyer yeni bir satış işlemi başlatır.
3. Kasiyer ürünün barkodunu girer.
4. Sistem bir satış kanalı maddesi oluşturur. Bu maddede ürün tanımı, fiyatı ve toplam bedel (aynı maldan birden fazla alınmış olabilir) yer alır.
5. Kasiyer 3. ve 4. adımları müşterinin alacağı tüm mallar için tekrarlar.
6. Sistem toplam bedeli vergi iadesi ile birlikte hesaplar.
7. Kasiyer müşteriye toplamı bildirir ve ödeme ister.
8. Müşteri ödemeyi yapar ve sistem ödemeyi tahsil eder.
9. Sistem tamamlanan işlemin kaydını tutmayı tamamlar ve harici envanter ile mali sistemlere gerekli bilgileri gönderir.
10. Sistem makbuz verir.
11. Müşteri mallarla birlikte ayrılır.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

KULLANIM SENARYOSU: Satış İşlemi

Alternatif Senaryolar:

3a. Geçersiz barkod

1. Sistem uyarı mesajı verir ve kayıt girişini reddeder.

3-7a. Müşteri bir kalem malı alışverişten çıkartmak ister.

1. Kasiyer satıştan çıkarmak üzere ürünün barkodunu okutur.
2. Sistem güncel toplamı bildirir.

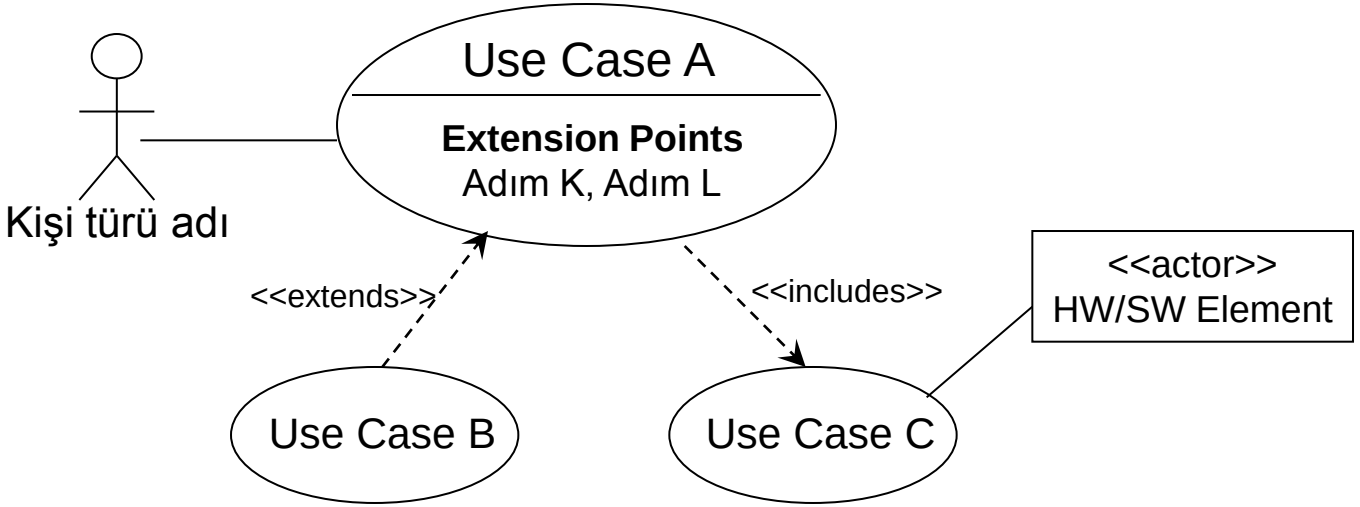
…

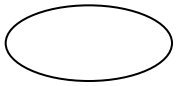
KULLANIM SENARYOLARININ GRAFİK GÖSTERİMİ

- Kullanım senaryoları, ayrıntılı ve uzun belgelerdir.
- Yazılımın yapacağı işlerin özet gösterimi için kullanım senaryosu şemaları kullanılır (use-case diagrams).
- Çizim kurallarını verdikten sonra örnek senaryonun şemasını çiz.

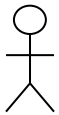
KULLANIM ŞEMALARI – USE CASE SCHEMAS

ÇİZİM KURALLARI

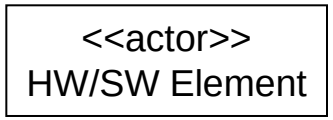




Use case:
İşlev

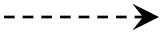


İnsan aktör:
Kullanıcılar



İnsan olmayan
aktör:

Etkileşim:
Kullanım



İlişki:
Benzeşim

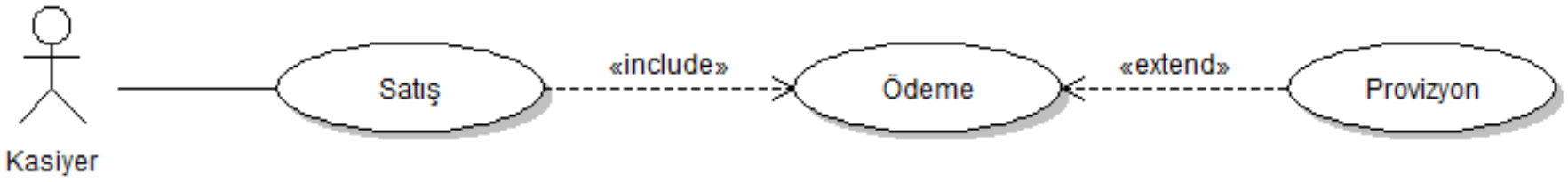
KULLANIM ŞEMALARI – USE CASE SCHEMAS

ÇİZİM BİLGİLERİ

- Benzeşim ilişkileri:
  - Ok yönü aynı zamanda ilişkiyi okuma yönüdür.
  - UC-B extends UC-A : B işlevi, A işlevi yürütülürken oluşabilecek bir sapış anlamındadır.
    - A: Ana akış
    - B: Ana akıştaki bir seçenek, ana akıştan bir sapış, alt akış
  - UC-A includes UC-C: A işlevi, C işlevini içerir.
    - A : Ana akış, içeren akış
    - C: Alt akış, içerilen akış

KULLANIM ŞEMALARI – USE CASE SCHEMAS

ÖRNEK ÇİZİM



- Bir POS yazılımının ödeme işlevini kasiyer kullanır.
- Satış işlevi, içerisinde ödeme yapma işlevini içerir.
  - Includes, çünkü: Her satış içerisinde mutlaka ödeme olur.
- Ödemenin kredi kartı ile olması halinde, provizyon alma işlemi yürütülür.
  - Extends, çünkü: Ödeme nakit ise provizyona gerek kalmaz.
  - Provizyon: Kredi kartının limitinin aşılıp aşılmadığı, çalıntı olup olmadığı, vb. gibi bilgilerin sınanması anlamında bir bankacılık terimi.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

SINIFLARIN BELİRLENMESİ

- Kullanıcı gereksinimleri belgesinden ve kullanım senaryolarından sınıfların elde edilmesi.
  - İsimlerin taranması
  - Sınıfların sağlaması gereken özellikler:
    1. Saklanan bilgi: Sistemin çalışması süresince bu varlığın durumu saklanmalıdır.
    2. Gereksinim duyulan hizmetler: Bu varlığın hizmetlerine ihtiyaç duyan başka varlıklar vardır.
    3. Çoklu nitelikler: Durum bilgisi birden fazla üye ile tarif edilmelidir.
    4. Ortak nitelikler ve/veya eylemler: Bir nesnenin üyelerinin tümü o nesnenin tüm türdeşleri için geçerli olmalıdır.
    5. Gerekli varlıklar: Problemin çözümü ile ilgili bilgi üreten veya problemin çözümü için bilgi tüketen varlıklar.
- Örnek gereksinim belgesinden sınıfları oluştur.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

SINIFLARIN BELİRLENMESİ

- Üyelerin belirlenmesi:
  - Sıfat ve eylemlerin taranması
  - Sorumlulukların belirlenmesi (CRC kartları)

- Sorumlulukların dağıtılması:
  - Sorumlulukların bir yerde yoğunlaşmaması
  - Sorumlulukların genelden özele doğru tanımlanması (kalıtım hiyerarşisinde genelden özele gidilmesi)
  - Bir bilgi ile ilgili davranışların, o bilgi ile aynı sınıfta yer alması (encapsulation)
  - Tek bir şey hakkındaki bilginin tek sınıfta yer alması
  - Gerekli sorumlulukların paylaşılması

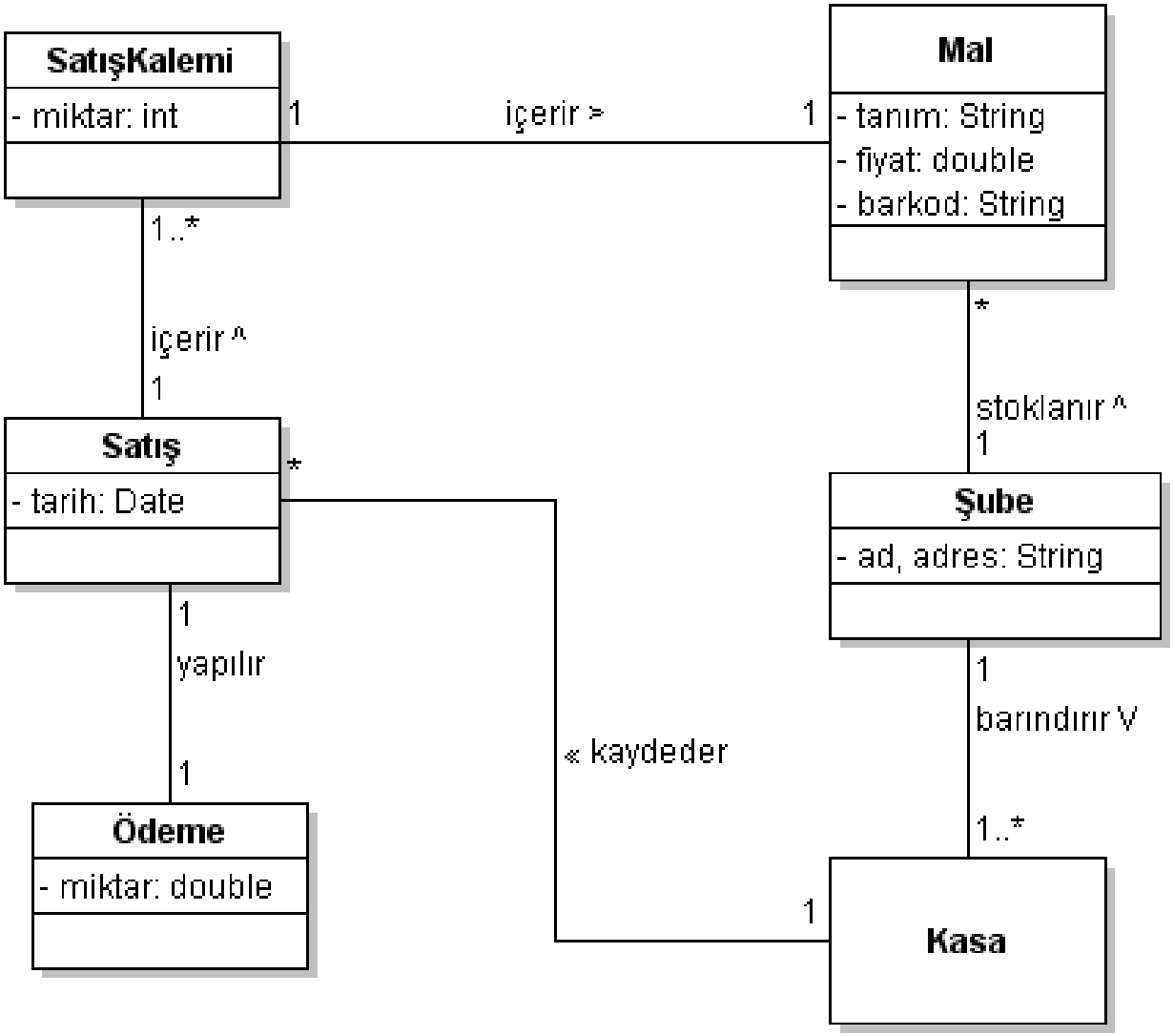
NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

ETKİLEŞİMLERİN BELİRLENMESİ

- Etkileşim: Bir nesnenin üzerine düşen sorumluluğu yerine getirmek için diğer bir nesneye mesaj göndermesi.
- Nesneler arasındaki ilişkiler
  - Bağlantı, toplama, meydana gelme.
- Sınıflar arasındaki ilişkiler
  - Özelleşme/genelleşme
- Çözümleme aşamasında ne tür etkileşimlerin olabileceği düşünülür, etkileşimlerin nasıl olacağı düşünülmez.

NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

ÖRNEK ALAN MODELİ



NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

ÇÖZÜMLEME SÜRECİNİN BELGELENDİRİLMESİ

- Bir nesneye yönelimli programın çözümleme sürecinin belgelendirilmesinde yer alan önemli belgeler:
  - UML Kullanım şemaları,
  - Kullanım senaryoları,
  - UML sınıf şemaları,
- Veritabanı işlemleri yapılacaksa bunlara ek olarak:
  - E-R diyagramı

NESNEYE YÖNELİK TASARIM SÜRECİ

GİRİŞ

- Nasıl? sorusuna yanıt aranır.
- Nesne modeli: Analizden tasarıma.
  - Doğrudan problem alanı ile ilgili nesnelerden oluşan model, yardımcı nesnelerle zenginleştirilir.
- Ana işlem grupları:
  - Nesne tasarımı: Problem alanı ile ilgili nesneler
  - Sistem tasarımı: Alt yapıyı ve gereçleri oluşturan nesneler
- Sistem katmanında bulunabilecek bileşenler:
  - Yazılım mimarisi: İstemci - sunucu, eşler arası, olay tabanlı, vb.
  - Kullanıcı arayüzü
  - Veri yönetimi
  - Ağ programlama
- Sistem katmanını çoğunlukla kendimiz sıfırdan oluşturmayız, hazır altyapı programlarını (framework) kullanırız.

NESNEYE YÖNELİK TASARIM SÜRECİ

TASARIM ÖLÇÜTLERİ

- Tasarım ölçütleri:
  - Ayrılabilirlik: Anlamlı parçalara ayrılabilme.
    - Parça: Sınıf/sınıf grubu.
    - Üstünde çalıştığımız problem hangi düzeyde alt problemlere bölünebiliyorsa, tasarımımız da aynı düzeyde ayrıştırılabilmelidir.
  - Birleştirilebilirlik: Bir parçanın başka tasarımlarda da kullanılabilecek şekilde diğer parçalarla birleştirilebilmesi.
  - Anlaşılabilirlik: Bir parçanın diğer parçalar hakkında bilgiye gerek duyulmadan anlaşılabilmesi.
  - Süreklilik: Yapılacak küçük değişikliklerin etkilerinin en az sayıda parçaya yayılması (tercihen tek sınıfa).
  - Koruma: Olası hataların düzeltilmesine yönelik büyük değişikliklerin etkilerinin geniş bir alana yayılmasının önlenmesi.

NESNEYE YÖNELİK TASARIM SÜRECİ

TASARIM İLKELERİ

- İyi bir tasarıma götüren iki temel ilke:
  - Düşük bağlaşım (Low coupling)
  - Yüksek uyum (High cohesion)

- Bu ilkeler hem birbirlerine hem de uygulama alanına bağımlıdır.
- Başka ilkeler de öne sürülebilir, ancak bu ikisi en temel ölçütlerdir.

NESNEYE YÖNELİK TASARIM SÜRECİ

DÜŞÜK BAĞLAŞIM – LOW COUPLING

- Bağlaşım: Bir parçanın diğer parçalara bağımlılık oranı.
  - Parça: Sınıf, alt sistem, paket
- Bağımlılık: Bir sınıfın diğerinin:
  - Hizmetlerinden yararlanması,
  - İç yapısından haberdar olması,
  - Çalışma prensiplerinden haberdar olması,
  - Özelleşmiş veya genelleşmiş hali olması (kalıtım ilişkisi).
- Çeşitli sınıf şemaları ile bağlaşım soruları sor.
  - İlişkide bulunulan diğer sınıfların sayısı arttıkça bağlaşım oranı artar.
- Düşük bağlaşımın yararları:
  - Bir sınıfta yapılan değişikliğin geri kalan sınıfların daha azını etkilemesi,
  - Yeniden kullanılabilirliğin artması

NESNEYE YÖNELİK TASARIM SÜRECİ

YÜKSEK UYUM – HIGH COHESION

- Uyum: Bir parçanın sorumluluklarının birbirleri ile uyumlu olma oranı.
- Yüksek uyumun yararları:
  - Sınıfın anlaşılma kolaylığı artar.
  - Yeniden kullanılabilirlik artar.
  - Bakım kolaylığı artar
  - Sınıfın değişikliklerden etkilenme olasılığı düşer.
- Genellikle:
  - Düşük bağlaşım getiren bir tasarım yüksek uyumu,
  - Yüksek bağlaşım getiren bir tasarım ise düşük uyumu beraberinde getirir.

NESNEYE YÖNELİK TASARIM SÜRECİ

TASARIMIN BELGELENDİRİLMESİ

- Bir nesneye yönelimli programın tasarım sürecinin belgelendirilmesinde yer alan önemli belgeler:
  - Ayrıntılı UML sınıf şemaları: Vazgeçilmez.
- Tasarımın ihtiyaçlarına göre alttaki diğer belgelerin çeşitli bileşimleri de kullanılabilir:
  - Sözleşmeler
  - UML Etkileşim şemaları (Interaction diagrams)
    - Sıralama şemaları (Sequence diagrams)
    - İşbirliği şemaları (Collaboration diagrams)
  - UML Etkinlik şemaları (Activity diagrams)
  - UML Durum diyagramları (State diagrams)

NESNEYE YÖNELİK TASARIM SÜRECİ

SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Sözleşme:
  - Kullanım senaryosunun ayrıntılandırılması ile elde edilir.
  - Bir nesnenin bir eylemi, bir sözleşme ile ayrıntılandırılır.
  - Her metota sözleşme yazılacak diye bir koşul yoktur.
    - Zaten kolay anlaşılabilir metotların sözleşmeye ihtiyacı yoktur.

NESNEYE YÖNELİK TASARIM SÜRECİ

SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Örnek sözleşme:

Sözleşme No: 2 – Satış Kalemi Girişi

İşlem:	ürünGir(barkod: Barkod, adet: integer)
Çapraz Başvurular:	Satış kullanım senaryosu
Ön Koşullar:	Süregelen bir satış işleminin olması
Son Koşullar:	- Bir SatışKalemi örneği olan satisKalemi oluşturulmuştur. - satisKalemi süregelen satis ile (Satış örneği) ilişkilendirilmiştir. - satisKalemi.adet üyesine o malın satış adedi atanmıştır. - satisKalemi, satılan mal ile uyuşan barkod sayesinde bir UrunTanimi örneği ile ilişkilendirilmiştir.

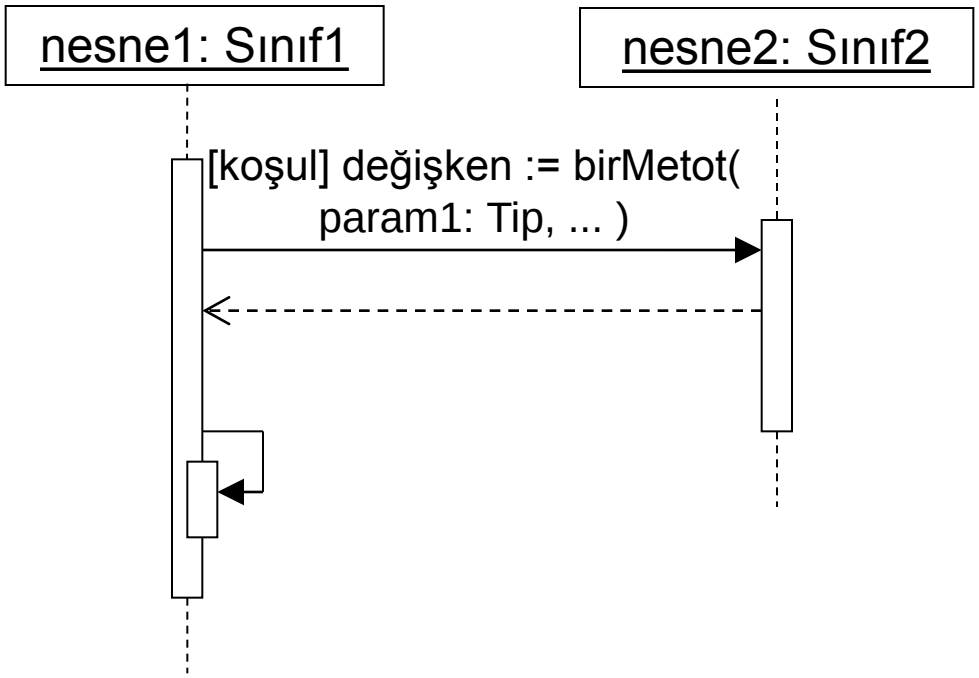
NESNEYE YÖNELİK TASARIM SÜRECİ

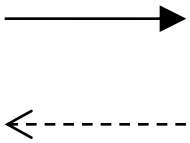
SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Sözleşmeler hakkında bazı ayrıntılar:
  - Ön koşullar tüm sistem hakkındaki bilgilerdir.
  - Son koşullar sadece problem alanı ile ilgili nesnelerin durum değişiklikleri hakkındadır.
  - Sözleşmeler her zaman gerekli olmayabilir.
  - Son koşullarda edilgen geçmiş zaman kullanılması, bunların işlemin sonunda tamamlanmış eylemler olduğunu vurgulamak açısından yerinde olacaktır.
  - Sözleşme içerisinde ilişkilerin kurulmasını belirtmeyi unutmayın!
  - Sözleşme yazılması problem alanı çözümlemesinde güncellemelere yol açabilir.

ETKİLEŞİM ŞEMALARI AYRINTILARI

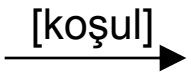
SIRALAMA ŞEMALARI



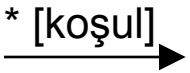


Mesaj (metot çağırma)

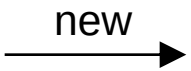
Geri dönüş (ihmal edilebilir)



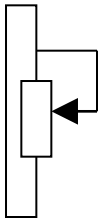
Koşullu mesaj (if)



Döngülü mesajlar (for, while, vb.)



Nesne oluşturma

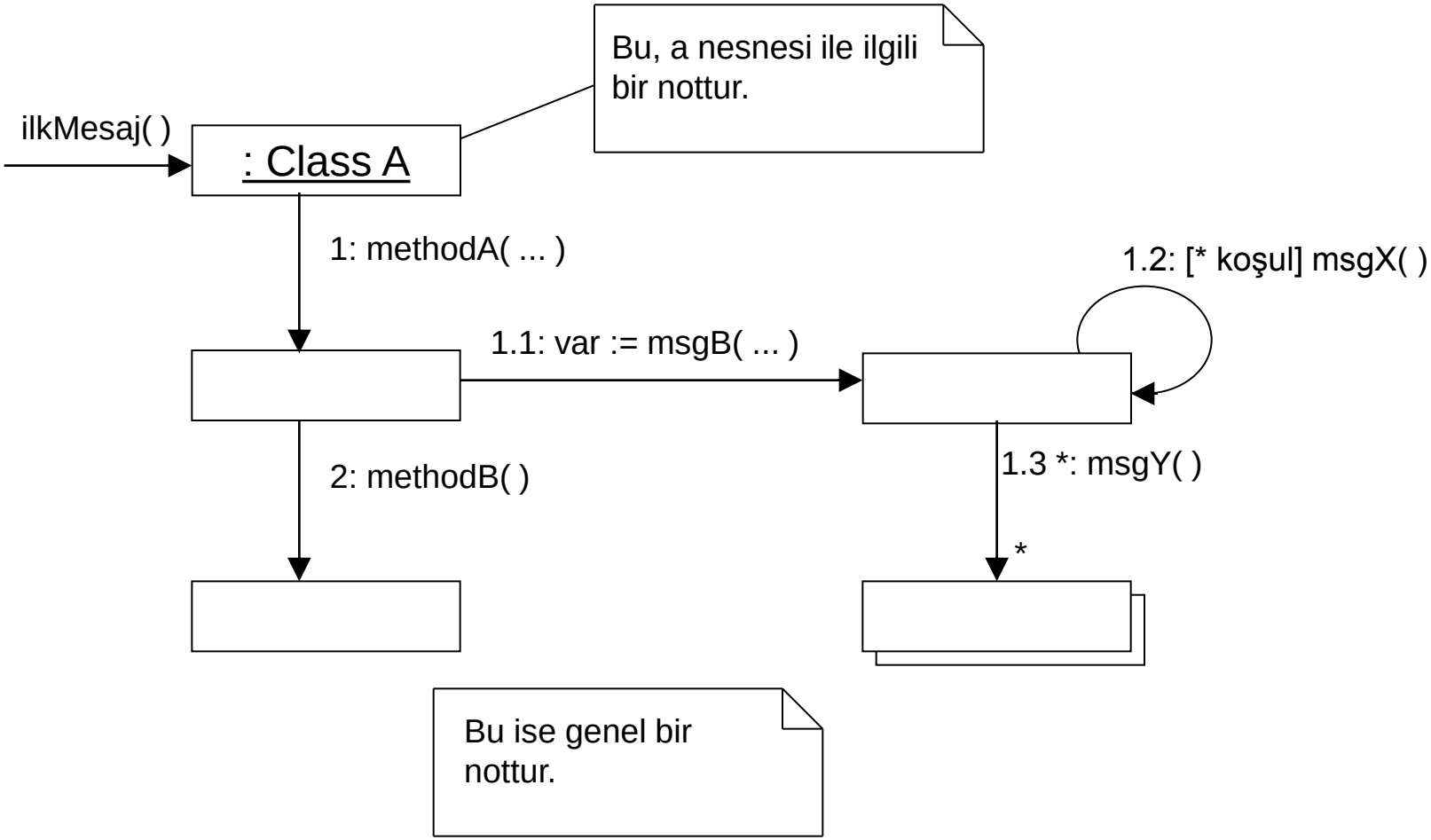


Nesnenin kendi metodunu çağırması

- Gizli bilgi:
  - birMetot Sınıf 2'ye aittir.
  - Sınıf 1'in bu çizimde gösterilmemiş başka bir metodunun gövdesinde nesne2.birMetot komutu verilmiştir.

ETKİLEŞİM ŞEMALARI AYRINTILARI

İŞBİRLİĞİ ŞEMALARI



ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

UYGULAMA ALANLARI

- Denetim akışını olaylar üzerinden göstermeye yarar.
- İş kurallarını göstermek ve paralel çalışma (multithreaded) ayrıntılarına girmek gerektiğinde, etkileşim şemalarından daha yararlıdır.

ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

ÇİZİM KURALLARI ve ÖRNEK ÇİZİM

Denetim akışı

Etkinlik başlangıcı

Sinyal alma /
Olay (event) kabul etme

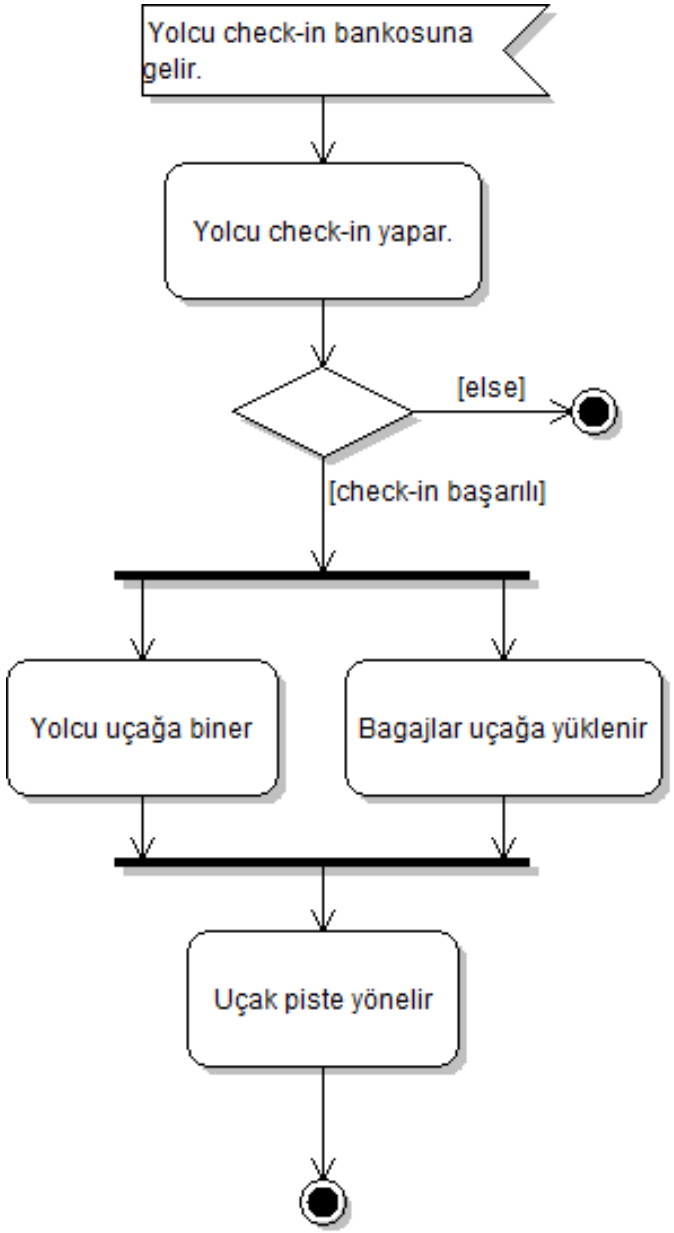
Eylem

Karar verme (if)

Eşgüdüm

Etkinlik sonu

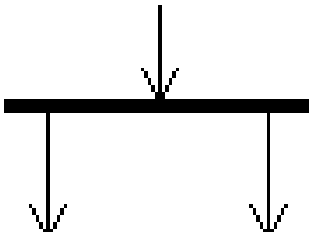
Sinyal gönderme

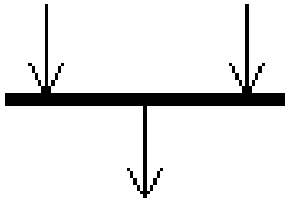


ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

ÇİZİM BİLGİLERİ

- Etkinlik şemaları başlangıç işareti veya sinyal alma işareti ile başlar.
- Sinyal alma: Beklemelidir.
  - Akış, bir sinyal alana kadar bekler.
  - Zamanlı olaylar da (timer) bununla gösterilebilir.
- Eşgüdüm: Beklemelidir.
  - Eşgüdüm çizgisine varan akış, çizgiyi geçmeden önce diğer akışların hepsini bekler.
    - fork join

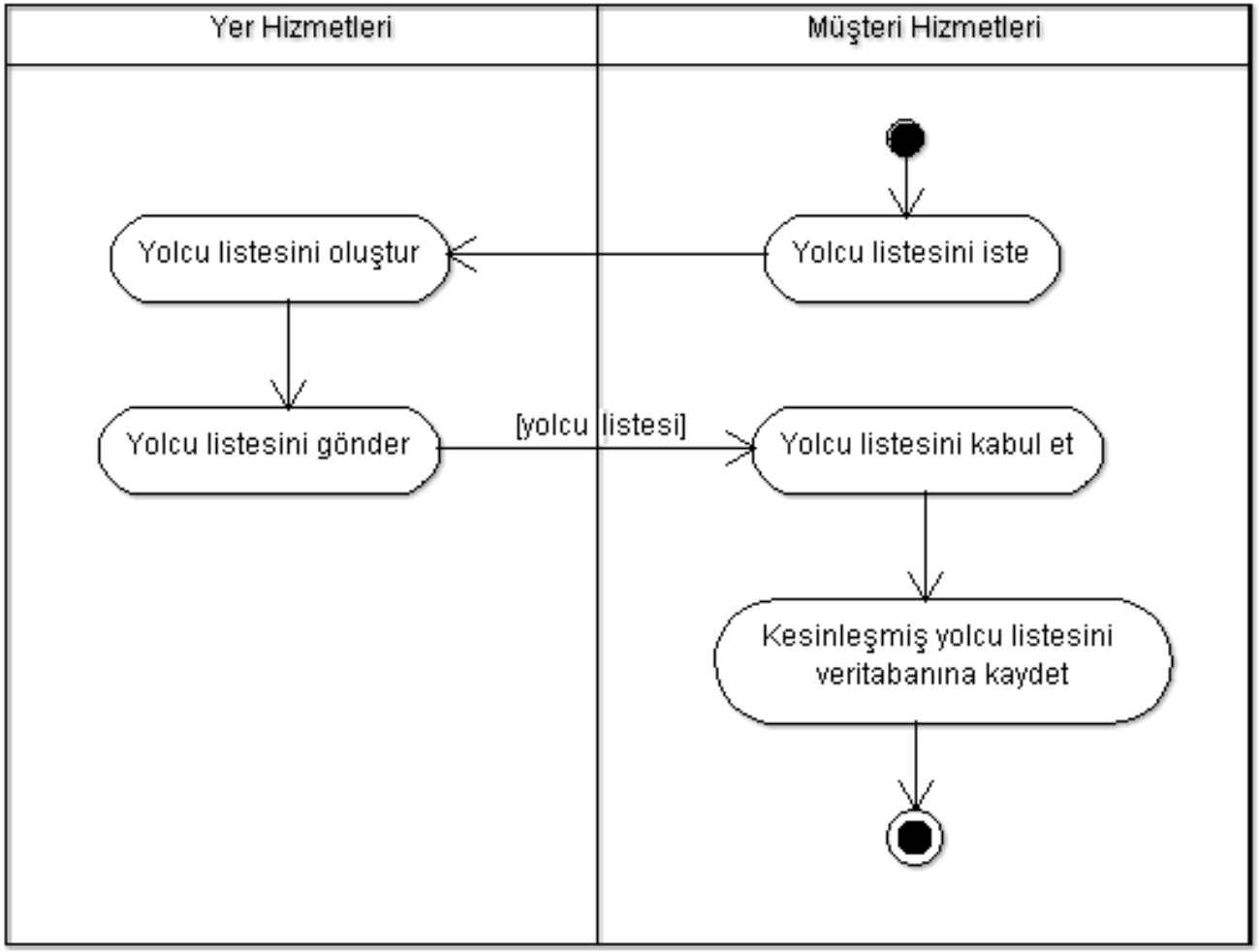




ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

ÖRNEK ÇİZİM

- Birden fazla aktörün ve aktörler arası bilgi akışının gösterilmesi:



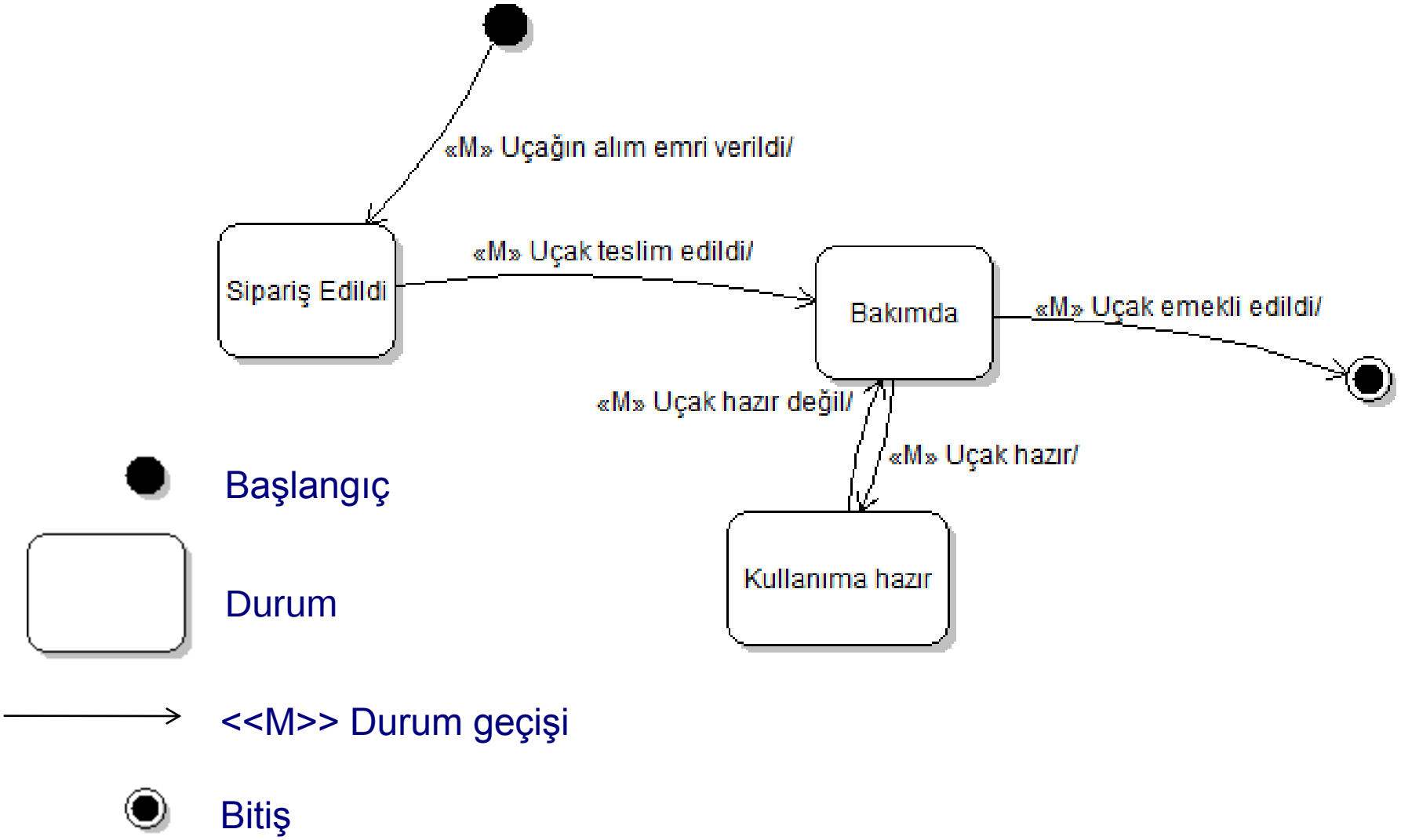
DURUM ŞEMALARI – STATE DIAGRAMS

UYGULAMA ALANLARI

- Bir varlığın içinde bulunabileceği durumları ve bu durumların birinden diğerine geçiş yapma kurallarını anlatmaya yarar.

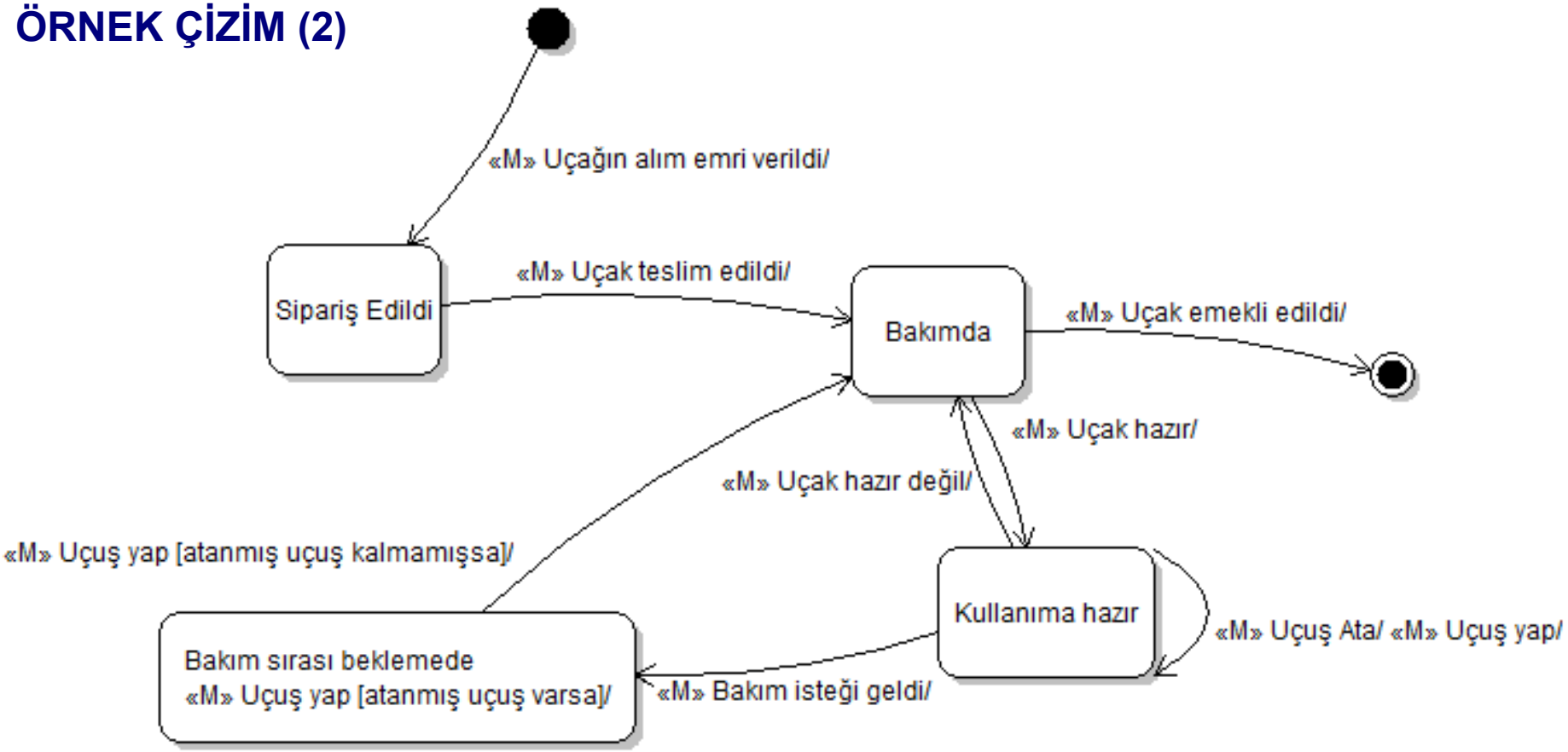
DURUM ŞEMALARI – STATE DIAGRAMS

ÇİZİM KURALLARI ve ÖRNEK ÇİZİM



DURUM ŞEMALARI – STATE DIAGRAMS

ÖRNEK ÇİZİM (2)



- Yorumlama: Kullanıma hazır bir uçak için bakım isteği gelmişse, uçak önce bakım sırasına alınır. Bu sırada önceden planlanmış uçuşları varsa onları yapar. Planlanan uçuşlar bitince uçak bakıma alınır.

DURUM ŞEMALARI – STATE DIAGRAMS

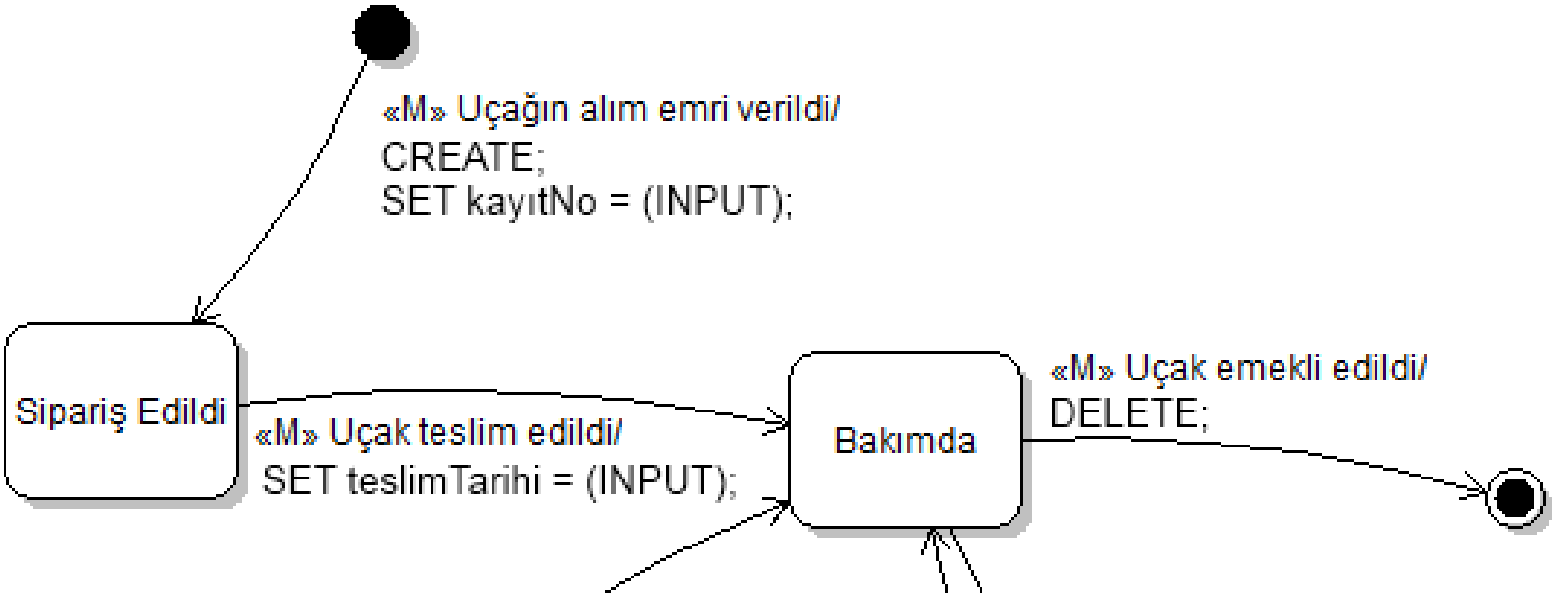
ÖRNEK ÇİZİM (2)

- Dezavantaj: Tutarlılık denetimi zor olabilir.
  - Karmaşık şemalarda mesajları takip etmek zorlaşır
  - Çünkü aynı mesaj birden fazla durumla ilişkili olabilir
    - Bu durumda aynı mesaj birden fazla yerde geçer.
  - Ör: Uçuş Yap mesajı.

DURUM ŞEMALARI – STATE DIAGRAMS

ÖRNEK ÇİZİM (3)

- Durum geçişi sırasında işlenen komutların şemada gösterilmesi:



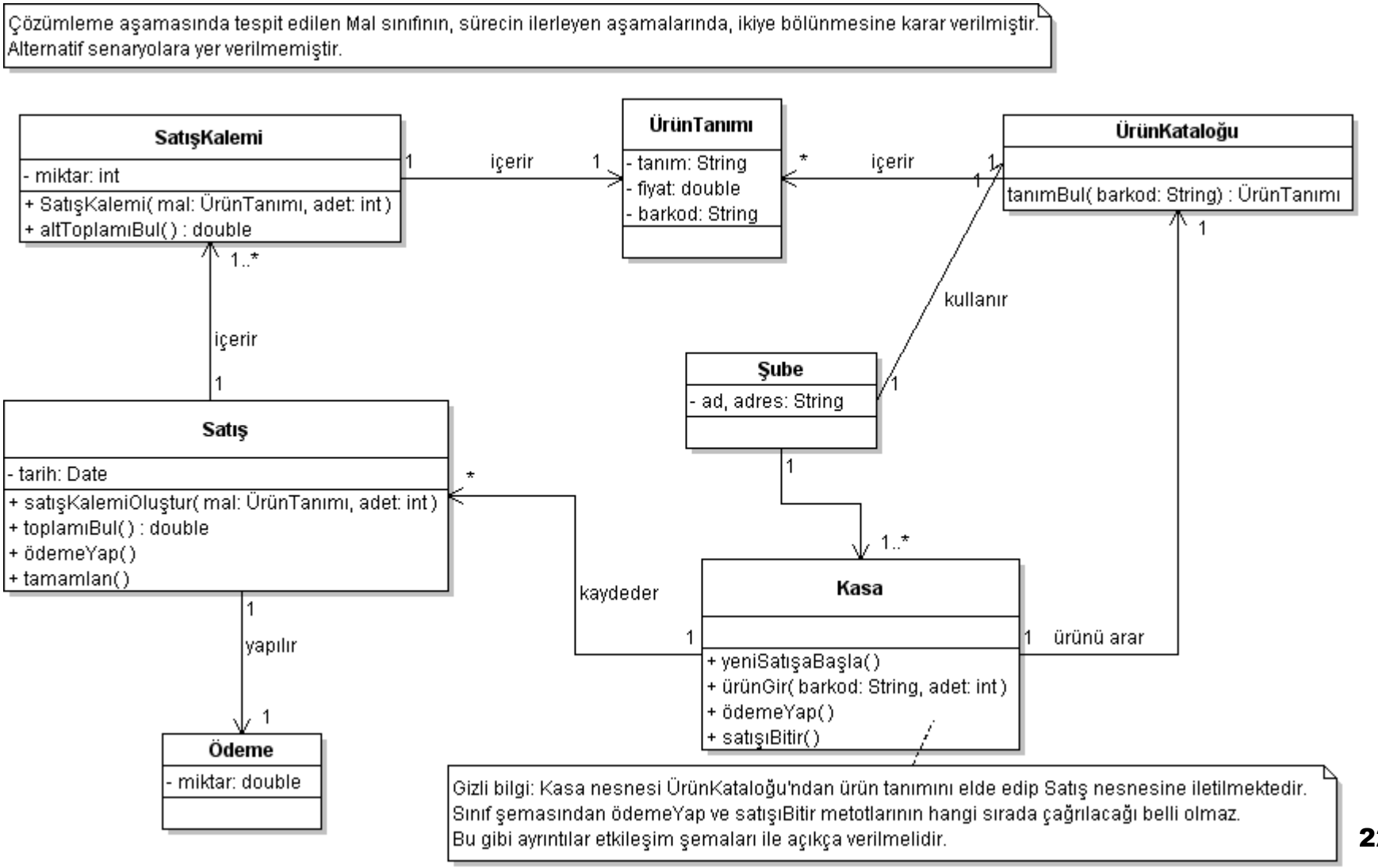
NESNEYE YÖNELİK TASARIM SÜRECİ

ANALİZDEN TASARIMA GEÇİŞ

- Satışı kasa üzerinden kasiyer başlatır, malların barkodlarını okutur, toplamı bildirir, ödemeyi alır ve satışı sonlandırır.
  - Bu işler nasıl olacak?
  - Alan modelini ayrıntılandırarak tasarımı gerçekleştirelim:

NESNEYE YÖNELİK TASARIM SÜRECİ

TASARIM MODELİ



YAZILIM SINAMA TEKNİKLERİ

GENEL BİLGİLER

- Yazılım geliştirme karmaşık bir süreç olduğundan, hataların ortaya çıkması kaçınılmazdır.
- Yazılım, yaşam döngüsünün <u>her aşamasında</u>, hatalara karşı sınanır.
  - Gereksinimler arasındaki tutarsızlıklar,
  - Çözümleme şeması ile uygulama alanı arasındaki uyumsuzluklar,
  - Tasarım hataları,
  - Çalışma anı hataları,
  - vb.
- Yaşam döngüsünde ilerledikçe, hataların düzeltilmesi zorlaşacaktır.
- İyi bir sınama yaklaşımı, hataların erken belirlenmesine katkıda bulunacak ve yazılımın kalitesini arttıracaktır.
- Bir sınama yaklaşımı şu bileşenleri içermelidir:
  - Planlama
  - Tasarım
  - Çalıştırma
  - Bilgi toplama ve değerlendirme

YAZILIM SINAMA TEKNİKLERİ

GENEL BİLGİLER

- Sınama yaklaşımlarının genel özellikleri:
  - Sınamalar bileşen düzeyinde başlar ve sistem düzeyinde sonlanır.
    - Ürün tek başına bilgisayar yazılımından ibaret olmayabilir.
    - Sistemin bütününde çeşitli algılayıcılar, harici sistemler, vb. yer alabilir.
    - NYP'de sınıfların bireysel sorumluluklarından başlayıp, yazılımın üst düzey sorumluluklarına kadar tüm işlevler sınanır.
  - Yazılım yaşam döngüsünün değişik aşamalarında değişik sınama teknikleri uygun olacaktır.
  - Projenin değişik aşamalarında değişik sınama teknikleri uygun olacaktır.
  - Sınamalar hem ilgili bileşeni oluşturan kişi tarafından, hem de bağımsız kişiler tarafından yapılır.
    - Büyük ölçekli kurumlarda ayrı bir sınama ekibi bulunabilir.
  - Sınama ve hata ayıklama ayrı işlerdir, ancak hata ayıklama her sınama yaklaşımının önemli bir parçasıdır.
  - Resmi teknik değerlendirmeler sayesinde sınama öncesinde de hatalar belirlenebilir.
    - Formal technical reviews, ileride değinilecek.

YAZILIM SINAMA TEKNİKLERİ

SINAMA AMAÇLI YAPILANMA

- Bir bileşen ilk olarak onu hazırlayan programcı tarafından sınanır.
  - Bir programı en iyi olarak bilen, onu yazandır.
- Her bileşen aynı zamanda yazarı dışındaki kişilerce sınanır.
  - Yazılım evinin bağımsız bir sınama ekibi bulunabilir (ITG: Independent Test Group).
    - Aksi halde farklı projelerde çalışan ekipler, veya aynı projenin farklı alt grupları, birbirlerinin çalışmalarını sınar.
  - Programcıların çıkarları, kendi yazdıkları programın hatasız, zamanında ve bütçe dahilinde tamamlanmasını gerektirir.
  - Psikolojik açıdan bakıldığında sınama ‘yıkıcı’ bir eylemdir.
  - Bu nedenlerle geliştiriciler kendi programlarının hatalarını bulmaya yönelik değil, doğru çalıştığını ispatlamaya yönelik sınamalara eğilim gösterebilir.
- Yazılım teknik olmayan kişiler tarafından da sınanır (ileride değinilecek).
- Sonuç:
  - Müşteriler hataları önünde sonunda bulacaklardır. İyisi mi siz onlardan önce davranın! Aksi halde prestijiniz sarsılacaktır.

YAZILIM SINAMA TEKNİKLERİ

SINAMA YAKLAŞIMININ BELİRLENMESİ

- Göz önünde bulundurulması gereken konular:
  - Sınama çalışmaları, en çok çaba gerektiren yazılım mühendisliği etkinlikleri arasında yer almaktadır.
  - Sınama çalışmaları, olası tüm hataların yakalanacağı bir ‘güvenlik ağı’ olarak düşünülmemelidir.
    - Sınama için ne kadar çaba gösterilirse gösterilsin, tespit edilemeyen hatalar olacaktır.
  - Hedefler açıkça belirlenmelidir:
    - Testlerin kapsama alanı,
    - Sınama çalışmalarına ne kadar kaynak ayrılacağı,
      - zaman, kişi, bütçe
    - Yazılımın türüne göre belirlenecek diğer çeşitli ölçütler
      - İki hata arası ortalama süre (MTBF), hassasiyet, vb.
  - Sınamalar otomatikleştirilmelidir.
  - Sınama süreci ölçülmeli ve iyileştirilmelidir.

YAZILIM SINAMA TEKNİKLERİ

SINAMA TÜRLERİ

- Yaklaşım tarzlarına göre sınama türleri:
  - Kara kutu sınaması (Black-box testing): Sınanacak birimin iç işleyişi bilinmez, sadece birimin beklenen girdilere karşı beklenen çıktıları üretip üretilmediğine bakılır.
  - Beyaz kutu sınaması (White-box testing): Sınanacak birimin iç işleyişi bilinir ve yapılacak sınamalar buna göre belirlenir.
- Yürütülme sıralarına göre sınama türleri:
  - Doğrulama Sınamaları (verification tests): Yazılım ekibi tarafından yapılır.
    - Birim sınamaları
    - Tümleştirme sınamaları
  - Geçerleme Sınamaları (validation tests): Son kullanıcılar tarafından yapılır.
    - Alfa sınaması
    - Beta sınaması

YAZILIM SINAMA TEKNİKLERİ

DOĞRULAMA SINAMALARI

- Birim sınamaları (Unit testing)
  - En küçük yazılım bileşeninin sınanmasıdır.
  - NYP'de bireysel sınıfların sınanmasıdır.
  - Ne zaman tasarlanır?
    - Kodlamadan önce (çevik yaklaşım),
    - kodlama sırasında,
    - veya kodlamanın ardından.
  - Kodlama sırasında veya kodlamanın ardından yürütülebilir.
  - Bir sınıfın tek başına yürütemediği sorumlulukların sınanması için, bu sınıfın ihtiyaç duyduğu diğer sınıfların yerine geçecek kod gerekebilir.
    - Vekil, sahte, yalancı kod/sınıf, vb.
    - Stub, dummy, surrogate, proxy, vb.
    - Vekil sınıflar, sadece ihtiyaç duyulan sınıflar gerçeklenene dek kullanılır.
    - Vekil sınıfların basit tutulması, ek kodlama yükünü azaltır.
      - Bu mümkün değilse, ortaklaşa yürütülen sorumlulukların sınanması tümleştirme sınamalarına bırakılır.

YAZILIM SINAMA TEKNİKLERİ

DOĞRULAMA SINAMALARI

- Birim sınamalarında aranabilecek hata türleri:
  - Farklı veri tiplerinin karşılaştırılması veya birbirinin yerine kullanımı.
    - NYP'de: Çokbiçimliliğin yan etkileri
  - Mantıksal işleçlerin yanlış kullanımı
  - İşleçlerin önceliklerinin gözden kaçırılması
  - Değişkenlerin karşılaştırılmasındaki hatalar
  - Döngülerin hatalı sonlanması veya sonsuz döngüler
  - Değişkenlere hatalı değerler atanması
  - vb.
- Yaptığınız hatalardan ders çıkarın:
  - Yapılabilecek tüm kodlama hataları öngörülemez, ancak kariyeriniz boyunca her hata yaptığınızda bu hatanızı 'aranabilecek hata türleri' listenize ekleyin.

YAZILIM SINAMA TEKNİKLERİ

DOĞRULAMA SINAMALARI

- Tümleştirme sınamaları (Integration testing)
  - Sınıflar birim sınamalarını geçmişlerse, bir araya getirildiklerinde de doğru çalışmazlar mı?
    - Yazılım geliştirme sürecinin her aşamasında her ayrıntının açıkça belirlenmesi beklenemez.
    - Ayrı ayrı programcılar, belirlenmemiş ayrıntılar üzerinde kendi karar verme yetkilerini (initiative) kullanabilir.
    - Aynı kişinin farklı ayrıntılar hakkında verdiği kararlar bile birbiri ile uyumlu olmayabilir.
  - NYP'de birim ve tümleştirme sınamalarını birbirinden ayıran kesin çizgiler yoktur.

YAZILIM SINAMA TEKNİKLERİ

DOĞRULAMA SINAMALARI

- Tümleştirme sınamaları türleri:
  - Tahribat sınaması (smoke testing)
    - Yüzeysel ancak başarısız olma durumunda tüm sistemin çalışmasının olanaksız olacağı sınamalardır (show-stopper errors).
    - Diğer tümleştirme sınamalarından önce yapılır.
    - Çeşitli bileşenler tüm gerekli yazılım elemanlarını içeren (kod, yapılandırma dosyaları, dış kütüphaneler, vb.) parçalar bir araya getirilir (build) ve günlük olarak sınanırlar.
  - Geriye dönük sınama (regression testing)
    - Yazılıma yeni bir işlev veya bileşen eklendiğinde, tüm sınamaların yenilenmesidir.
    - Hangi ölçekte bir eklentinin geriye dönük sınamayı başlatacağının kararını vermek gerekir.
      - Ölçek düştükçe sınama sıklaşır ve masraf artar.
  - Otomatik sınama gereçleri kullanılarak masraflar azaltılabilir.

YAZILIM SINAMA TEKNİKLERİ

GEÇERLEME SINAMALARI

- Geçerleme sınamaları (validation testing):
  - Gereksinimler belgesinde yazılmış olan işlevsellikten yola çıkılarak, kullanıcı tarafından yapılır.
  - Son kullanıcıların yapabileceği beklenmedik davranışların tümünü, teknik ekip önceden bilemez.
    - ‘Fincan tutacağının çalışmaması’, ‘Pencereyi açmak’, …
  - Bir şeyi ne kadar çok kişi incelerse, ondaki kusurlar o kadar çabuk bulunur ve düzeltilir.
    - Eric Raymond: “given enough eyeballs, all bugs are shallow”
  - Alfa ve Beta sınaması olmak üzere iki türü vardır.

YAZILIM SINAMA TEKNİKLERİ

GEÇERLEME SINAMALARI

- Alfa sınaması:
  - Yazılım firması içerisinde, kullanıcı tarafından yapılır.
  - Yazılım geliştirme ekibinin denetiminde ve izlemesi ile yapılır.
  - Yazılımın doğal kullanım ortamına en yakın koşullarda yürütülür.
- Beta sınaması:
  - Müşterinin kendi yeri içerisinde, gerçek kullanım ortamında yapılır.
  - Yazılım geliştirme ekibi müdahil olmaz.
  - Bulunan hatalar yazılım geliştirme ekibine düzenli aralıklarla ve resmi bir biçimde bildirilir.

YAZILIM SINAMA TEKNİKLERİ

SİSTEM SINAMALARI

- Yazılım tek başına sistemin bütününü oluşturmayabilir.
  - Gömülü uygulamalar, ara katman yazılımları, vb.
- Sistemin tümünü her yönüyle incelemeye yönelik sınamalardır:
  - Kurtarma (recovery) sınaması:
    - Hatalara dayanıklı (fault tolerant) sistemler için geçerlidir.
    - Bir hata ortaya çıktığında sistemin kendini toparlayarak doğru çalışmaya devam edip edemediği sınanır.
    - Kurtarma işlemi belli bir süre içerisinde tamamlanmalıdır (MTTR: Mean Time To Repair).
  - Güvenlik (security) sınaması:
    - Tek kural: Kural yok!
    - Her güvenlik önünde sonunda aşılır!
  - Zorlama (stress) sınaması:
    - Normalin dışında yüklenme durumunda, sistemin nereye kadar dayanabileceğinin sınanması
  - Başarım (performance) sınaması:
    - Gerçek zamanlı uygulamalarda özel öneme sahiptir.
    - Program kendisinden bekleneni doğru yapabilir ama zamanında yapamayabilir.

YAZILIM SINAMA TEKNİKLERİ

HATA AYIKLAMA (DEBUGGING)

- Yapılan türlü sınamaların sonucunda bulunan hatalar düzeltilmelidir.
- Sınama ve hata ayıklama çalışmaları kimileri tarafından ‘angarya’, ‘ayak işi’, ‘sıkıcı’, ‘ikinci sınıf’ olarak nitelendirilebilir.
  - Sınamanın önemini gördük
  - Yazılımınızdaki hatalar size ve kurumunuza prestij kaybettirir.
  - Düzeltilmeyen veya düzeltmesi uzun süren hatalar ise daha çok prestij kaybettirir.
  - Sınama ve güvenlik açıklarının belirlenmesi için alışılagelmiş düşünce biçiminin dışına çıkabilmek gerekir ki bu da özel bir yetenektir.
  - Saygı duyulan ve ünlü programcılar, hata ayıklamanın kodlama yapmaktan daha zor ve daha çok yetenek isteyen bir iş olduğu görüşünde birleşmektedirler.
- Bir noktada takıldığınızda, biraz ara verip sorunu yeniden incelemeniz, başarı şansınızı arttıracaktır.
- Geliştirme ortamının hata ayıklama yeteneklerinden sonuna kadar yararlanın.

YAZILIM SINAMA TEKNİKLERİ

YAZILIM GÖZDEN GEÇİRME EYLEMİ

- Yazılım gözden geçirme: Software review
  - Yazılım sınamasından önce, sınama eylemlerinden daha az masrafla, yazılım hatalarının bulunmasını amaçlar.
  - Sınama çalışmaları ile bulunabilecek tüm hatalar gözden geçirme ile bulunamaz, ancak gözden geçirme daha verimlidir.
  - Gözden geçirme resmi veya gayri resmi olabilir.
    - Çalışmalar resmi gözden geçirmelerin daha etkili olduğunu göstermiştir.
      - Gözden geçirme toplantılarının yapılması durumunda.
    - Çevik süreçlerden XP'deki eşli programlama da bir tür gözden geçirmedir.
- Resmi yazılım gözden geçirme: Formal software review
  - Toplantı şeklinde yapılır.
  - Toplantının bir yöneticisi (review leader) bulunur.
  - Bu konuda çeşitli standartlar önerilmiştir.
    - Ör: IEEE 1028 standartı

YAZILIM SINAMA TEKNİKLERİ

RESMİ YAZILIM GÖZDEN GEÇİRME ÇALIŞMALARI

- IEEE 1028 std. göre bir resmi yazılım gözden geçirme adımları:
  - Değerlendirme başlangıcı: Değerlendirme yöneticisi standart bir checklist kullanarak, verimli bir toplantı için gerekli koşulları sağlar.
  - Yönetimin hazırlanması: Sorumlu yönetim gözden geçirme için gerekli kaynakları hazırlar ve toplantının standartlara uygun yürütülmesini sağlar.
  - Gözden geçirme prosedürlerine genel bakış: Değerlendirme yöneticisi tüm değerlendiricilerin gözden geçirmenin amaçlarını ve prosedürlerini anladığından emin olur.
  - Bireysel hazırlık: Değerlendiriciler bireysel olarak inceleme toplantısına hazırlanır. Bu amaçla gözden geçirilecek malzemede hataya yol açabilecek olası bozukluklar (anomaly) aranır.
  - Grup incelemesi: Bireysel hazırlıkların sonuçları önceden belirlenen yer ve zamandaki toplantıda biraraya getirilir ve sonuç raporu üzerinde uzlaşıya varılır.
  - Düzenleme: İncelenen çalışmanın yazar(lar)ı veya atanacak bir başka kişi/ekip, önceki adımda belirlenen noktaları düzeltir.
  - Sonlandırma: Değerlendirme yöneticisi düzeltmelerin yeterliliğini inceler.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

SÜREÇ MODELLERİ: SÜREÇ İYİLEŞTİRME VE SERTİFİKASYONU

- Amaç: Yazılım sürecini ve proje yönetimini iyileştirerek kaliteyi arttırmak.
- İçerik:
  - Yazılım süreci içerisinde yapılması önerilen işler.
  - Yazılım sürecinin sahip olması gereken nitelikler.
- Nitelikler, işleri dikte etmez!

- Süreç Modelleri veya standartları:
  - CMMI: Capability Maturity Model Integration
  - ISO 9001:2000 standartları.
  - PMI

- Süreç modelleri birbirleri ile tam örtüşmez:
  - Aynı firma bir modelden yüksek not alıp diğerinden düşük not alabilir.

- Ordu, istihbarat, bakanlıklar gibi kurumlar ile alt yüklenici arayan özel şirketler, çalışacakları firmaların süreç modellerinin birinden belli bir derecenin üstünde not almalarını şart koşabilir.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

CMMI SÜREÇ MODELİ

- Carnegie-Mellon Universitesi'nin yazılım müh. enstitüsü tarafından önerilmiştir (SEI: Software Engineering Institute)
- CMMI düzeyleri:
  - 1. Düzey: Giriş düzeyi (Level 1: Initial). İş şansa ve anahtar kişilere kalmış.
  - 2. Düzey: Yinelenebilir (Repeatable). Temel planlama ve izleme yöntemleri kullanılarak, önceki projelerdeki başarılar yeni projelerde tekrarlanılabilir.
  - 3. Düzey: Tanımlanmış (Defined). Kişi ve risk yönetimi ile projenin yönetimi iyileştirilir.
  - 4. Düzey: Yönetilen (Managed). Süreç ve yazılım ölçütleri kullanılarak kalite yönetimine geçilir. İlerleme sürekli izlenir, bütçe ve zaman hedeflerinden sapmalar erkenden belirlenerek gerekli önlemler alınır.
  - 5. Düzey: İyileştirilmiş (Optimized). Süreç yönetimi geçmiş deneyimlerin ışığında sürekli iyileştirilir.
- 700'den fazla sayfaya sahip dokümanı için: http://www.sei.cmu.edu/cmmi/

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

CMMI SÜREÇ MODELİ

- CMMI, 25 süreç alanına sahiptir.
- CMMI iki şekilde tanımlanabilir veya gerçeklenebilir:
  - Aşamalı (Staged): Tüm alanlarda mümkün olan aynı en üst düzeye çıkmak (düzey:1-5).
  - Sürekli: Farklı alanlarda farklı düzeylerde bulunmak.

YAZILIM YAŞAM DÖNGÜSÜ VE YAZILIM GELİŞTİRME SÜREÇLERİ

DİĞER OLGUNLUK SÜREÇLERİ

- ISO 9001:2000 – Yazılım mühendisliğine yönelik süreç iyileştirme standartları.
- PMI: Her tür projeye uyarlanabilir, Bilişim projeleri alt dalı da mevcut.

SÜREÇ MODELLERİ: GERÇEKTEN GEREKLİ Mİ?

- Gerekli olduğu durumlar:
  - Büyük şirketler ve büyük projeler için uygundurlar.
  - Bazı müşteriler süreç modeli isteyebilir.
    - TSK ve Bakanlıklar: CMMI Düzey 3
- Gereksiz olduğu durumlar:
  - Küçük şirketler ve küçük projeler için ek yük olarak görülebilir.
  - Şirket kültürü süreç modellerine uygun olmayabilir.
- Sonuç:
  - Süreç modellerinin tüm eylemleri kullanılmasa bile, süreç modellerinin ruhu, takımın ruhuna işlemelidir.
  - En azından çevik bir süreç kullanın.

YAZILIM PROJE YÖNETİMİNE GİRİŞ

GENEL BİLGİLER

- Yazılım projeleri önemli oranda başarısızlığa uğramaktadır:
  - Yazılım geliştirmedeki zorluklar.
  - Ölçek büyüklüğünden kaynaklanan zorluklar: Yazılım ölçeği, kişi ölçeği, vb.
  - Kestirimdeki zorluklar.
  - İnsanlarla çalışmadaki zorluklar.
  - Teknolojideki değişimler.
  - Gereksinimlerdeki değişimler.
  - Politik değişimler.
  - Mali değişimler.
- Yazılım proje yönetimi, sayılan zorlukların çözümüne odaklanır.
- Önem sırasına göre proje yönetiminin ilgi alanları:
  - Kişiler
  - Ürün
  - Süreç
  - Proje

YAZILIM PROJE YÖNETİMİNE GİRİŞ

GENEL BİLGİLER

- Temel ilkeler:
  - Temel amacın kullanıcılara bir yarar sağlamak olduğunu hiçbir aşamada unutmayın.
  - Geleceğin getireceği değişikliklere açık olun – ancak çok fazla açılıp da boğulmayın.
  - Yeniden kullanım sürekli aklınızda olsun.
  - Düşünün!

- Planlama yaklaşımları:
  - Basitlik yaklaşımı: Geleceğin getireceği değişiklikler çoğu zaman ayrıntılı planlama gereğini ortadan kaldırır.
  - Geleneksel yaklaşım: Planlama proje için bir yol haritası belirler; harita ne kadar ayrıntılı ise kaybolma olasılığı o kadar düşer.
  - Çevik yaklaşım: Ön hazırlık gereklidir ancak asıl harita proje ilerledikçe çizilir.

YAZILIM PROJE YÖNETİMİNE GİRİŞ

PLANLAMA

- Planlama ilkeleri:
  - Projenin sınırlarını belirleyin: Nereye gideceğinizi bilmezseniz kaybolursunuz.
  - Müşteriyi planlama eylemlerine katın: Öncelikleri, sınırları ve zamanlamayı müşteri belirler (bu sırada) ancak gerçekçiliği korumak amacıyla yazılım ekibi pazarlık yapar (aksi sırada).
  - Planlamanın doğası yinelemelidir: Plan asla taşa yazılmaz.
  - Bildiklerinizi kullanarak kestirimlerde bulunun: Bilginin kapsamı, doğruluğu ve belirginliği kestirimin doğruluğunu etkiler.
  - Planın her aşamasına risk değerlendirmesini ekleyin: Teknik, mali, kişisel, politik riskler.
  - Gerçekçi olun: Yazılımcı süpermen veya robot değildir.

YAZILIM PROJE YÖNETİMİNE GİRİŞ

PLANLAMA

- Planlama ilkeleri (devam):
  - Planların ölçeği: İnce ayrıntılı planlar daha kısa vadeli, genel ayrıntılı planlar daha uzun vadelidir. Zaman ilerledikçe genel ayrıntıdan ince ayrıntıya geçilir.
  - Kalite güvence eylemlerini tanımlayarak plana ekleyin.
  - Değişikliğin nasıl kabul edileceğini müşteri ile sözleşmeye bağlayın.
  - Planın gidişinden gözünüzü ayırmayın ve gerekli ayarlamaları yapın.

YAZILIM PROJE YÖNETİMİNE GİRİŞ

GEREKSİNİM MÜHENDİSLİĞİ

- Üzerinde çalışılmaya başlanacak projenin amaçlarını, boyutlarını ve etkilerini, özetle gereksinimlerini ve yapılabilirliğini (feasibility) belirlemeye yönelik çalışmalardır.
- SORU: Projenin boyutları neler olabilir?
  - Parasal (bütçe)
  - İnsan kaynakları
  - Zamanlama
- Müşteri ne istediğini bilmez mi? Gereksinimler zaten belli değil mi?
  - Çoğunlukla müşterinin kafasında sadece genel bir fikir vardır.
  - Yoruma açık ve ayrıntıları kesin çizgilerle belirlenmemiş gereksinimler projenin başarısızlığına davetiye çıkarır.
  - Kesin belirlenmiş gereksinimler bile zaman içerisinde değişebilir.
- Deyişler:
  - Şeytan ayrıntıda gizlidir.
  - Yanlış veya eksik işi yapan mükemmel yazılım değil, doğru işi yapan iyi çözüm gereklidir.
- SONUÇ: Gereksinim mühendisliği gerekli bir etkinliktir.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Gereksinim mühendisliğinin genel adımları:
  - Başlangıç (Inception)
  - Bilgi Toplama (Elicitation)
  - İşleme (Elaboration)
  - Pazarlık (Negotiation)
  - Tanımlama (Specification)
  - Doğrulama (Validation)
  - Yönetim (Management)
- Gereksinim mühendisliği adımları gerçeklenecek yazılımın doğasına ve kullanılan sürece göre düzenlenmelidir.
- Gereksinim mühendisliği adımları süresince yazılım ekibi ve müşteri birlikte çalışmalıdır.
  - Müşterinin bir ekibinin, yazılım geliştirme sürecinin mümkün olduğunca çok adımının bir parçası olması yararlıdır.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Başlangıç:
  - Yazılım projesinin **<u>ilk aşamalarının</u>** başlatılıp başlatılmamasına karar verilen adımdır.
  - Müşterinin bir yazılım projesi başlatılmasını düşünmesine neden olan olaylar:
    - Yeni bir iş gereksiniminin belirlenmesi.
    - Mevcut iş süreçlerinde güçlüklerle karşılaşılması.
    - Müşterinin üst düzey karar vericileri ve astları arasında geçen kısa bir sözlü konuşma veya toplantı ile bile bir proje başlayabilir.
  - Bir uygulama yazılımı söz konusu ise:
    - Yeni bir pazarın veya hizmetin farkına varılması,
    - Yazılım şirketinin üst düzey karar vericileri ve teknik ekibinin sözlü konuşması ile yeni bir yazılım projesi başlatılabilir.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Başlangıç aşamasında paydaşlar belirlenmelidir.
  - Paydaş: Gerçeklenecek sistemden doğrudan veya dolaylı olarak yararlanabilecek ve etkilenebilecek herkes.
    - Her paydaş sisteme farklı bir açıdan bakar.
    - Projenin başarısı veya başarısızlığı paydaşları farklı şekillerde etkiler.
    - Paydaşlara sorulacak sorularla belirlenmesi gerekenler:
      - Paydaşların bakış açıları,
      - Paydaşları etkileyebilecek nedenler,
      - Söz konusu etkilerin sonuçları.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Bilgi toplama aşamasının genel ilkeleri:
  - Gereksinimler hakkında ayrıntılı bilgiler, tüm paydaşların etkin katılımı ile elde edilmelidir.
  - Tüm paydaşların katıldığı toplantılar yapılmalıdır.
  - Toplantılara hazırlık ve katılım kuralları belirlenmelidir.
  - Gündem belirlenmelidir: Önemli konuları atlamayacak kadar sıkı, yaratıcılığı önlemeyecek kadar açık olmalıdır.
  - Düzeni sağlayacak ve tıkanıklıkları çözecek bir oturum başkanı seçilir.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- İşleme:
  - Bilgi toplama aşamasında toplanan ‘ham’ bilgilerin ‘işlenmesi’.
  - Son kullanıcının ve diğer paydaşların yazılımla nasıl etkileşimde bulunacağının belirlenmesi ve ayrıntılandırılmasını amaçlar.
  - Etkileşimler, kullanım senaryoları ile gösterilir (ileride anlatılacak).
  - İşleme kimi bilgilerin genişletilmesi, kimi bilgilerin özetlenmesi şeklinde gerçekleşir.
  - Gereksinimlerin sınıflandırılması
    - Normal gereksinimler
    - Beklenen gereksinimler: Çok temel gereksinimleri kullanıcı belirtmeyebilir. Bunların da elde edilmesi gereklidir.
    - Heveslendirici gereksinimler: Müşteri beklentilerinin ötesinde ve varlığında müşteriyi sevindirecek özellikler.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Pazarlık:
  - Müşteriler sınırlı insan, zaman ve bütçe kaynakları çerçevesinde karşılanamayacak aşırı isteklerde bulunabilir.
  - Paydaşlar gereksinimleri farklı önem düzeylerinde görebilir.
  - Farklı paydaşların gereksinimleri birbiri ile çelişebilir.
  - Pazarlık sonucunda tüm paydaşların razı olacağı bir gereksinimler listesi elde edilir.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Tanımlama:
  - Gereksinimler tanımlama aşamasında, pazarlık sonucu üzerinde uzlaşılan haliyle kağıda dökülür.
  - Tanımlama araçları:
    - Konuşma dili ile yazılmış belgeler
    - Kullanıcı senaryoları: Görülecek
    - Kullanım şemaları: Görülecek
    - Formel modeller (Matematiksel gösterim, işlenilmeyecek)
    - Bir ön ürün
  - Birden fazla tanımlama aracı birlikte kullanılabilir.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Doğrulama:
  - Tanımlanmış gereksinimlerin tutarsızlıklara karşı sağlaması yapılır.
    - Gereksinimler açıkça ve yoruma yer bırakmayacak şekilde tanımlanmış mı?
    - Birbiri ile çelişen gereksinimler var mı?
    - Gereksinimlerde hatalar ve eksikler var mı?
    - Eksik gereksinimler var mı?
    - Gerçekçi olmayan gereksinimler var mı?
    - …
  - Doğrulama yapma için önerilen temel yol teknik değerlendirmedir (Formal technical review, anlatılmayacak).

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Yönetim:
  - Yazılım geliştirme süreci içerisinde gereksinimlerde değişiklikler olabilir:
    - Yeni gereksinimler eklenmesi
    - Mevcut gereksinimlerden bazılarının geçerliliğini yitirmesi
    - Gereksinimlerin önem sıralamasının değişmesi
    - Hatalı kestirimlerden dolayı bazı gereksinimlerden vazgeçilmesi
  - Gereksinimlerde ne tür değişikliklerin nasıl ve hangi şartlarla yapılabileceği, resmi bir sözleşme ile önceden belirlenebilir.
  - Gereksinimlerde değişiklikler müşteri ile karşılıklı anlaşma ile yapılmalıdır.

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Yönetim (devam):
  - Yazılım geliştirme süreci içerisinde gereksinimlerin gerçeklenmesinin (ve varsa gereksinimlerdeki değişikliklerin) izlenmesi gerekir.
  - İzleme tablolar aracılığı ile yapılır:
    - İzlenebilirlik tabloları (Tracebility table).

	B1	B2	B3	…
G1	✓		✓	
G2		✓		
G3	✓			
…				

- G1,2,…: Gereksinimler
- B1,2,…: Sisteme çeşitli bakış açıları

GEREKSİNİM MÜHENDİSLİĞİ

GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)

- Yönetim (devam):
  - İzlenecek eşleşmeler (bakış açıları):
    - Gereksinimlerin birbirleri ile ilişkileri,
    - Gereksinimlerin yazılımın çeşitli bakış açıları ile olan ilişkileri,
    - Gereksinimlerin yazılımın içinde bulunduğu sistemin bileşenleri ile ilişkileri.

	B1	B2	B3	…
G1	✓		✓	
G2		✓		
G3	✓			
…				

- G1,2,…: Gereksinimler
- B1,2,…: Sisteme çeşitli bakış açıları

YMH 114 Stratejik Bakış

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

YMH 114 için Stratejik Bakış-Sunum İçeriği

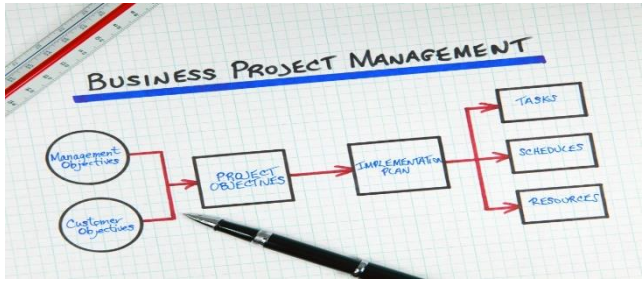
- Proje Nedir?
- Proje Neleri Sağlamalıdır?
- Proje-Fikir İlişkisi: Her Fikir Proje midir?
- Amaç Nedir? Amaç neleri içermelidir?
- Sorun Analizi ve Hedef Analizi ne demektir?
- Araştırma Projesi Nedir?
- Proje seçiminde öneriler?

Proje Nedir?

“Proje” belirli bir yerde, belirli bir zaman ve bütçe çerçevesinde, bir başlama ve bitiş noktasına sahip, önceden belirlenmiş amaçlara ulaşılmasını sağlayacak olan faaliyetler topluluğudur.

Proje Neleri Sağlamalıdır?

- Belirli bir yerde uygulanmalı,
- Takvimi olmalı,
- Bütçesi olmalı,
- Tanımlanmış faaliyetleri olmalıdır.



Proje-Fikir İlişkisi: Her Fikir Proje midir?

Her fikir bir proje midir?

Hayır, değildir. Bir fikrin proje olabilmesi için aşağıdaki sorulara cevap verebilecek olgunluğa erişmiş olması gerekmektedir.

- ♦ Nerede uygulayacaksınız?
- ♦ Ne kadar sürede bitireceksiniz?
- ♦ Belirlediğiniz yer ve sürede gerçekleştireceğiniz faaliyetler için ne kadar mali kaynağa ihtiyacınız var?
- ♦ Belirlediğiniz yerde ve sürede hangi faaliyetleri gerçekleştireceksiniz?

Amaç Nedir? Amaç neleri içermelidir?

- Amaçlarınız projeniz ile ortadan kaldırmayı, iyileştirmeyi önerdiğiniz sorunlara işaret etmelidir. Dolayısı ile projenin genel amacı ve özel amacını/amaçlarını doğru ve detaylı bir şekilde ortaya koyabilmek, tekrarlara düşmemek adına **sorun analizi** ve **hedef analizi** yapmanız önerilir.

Sorun Analizi ve Hedef Analizi ne demektir?

Sorun analizi, projenin çözmekte rol oynayacağı/katkı sağlayacağı temel sorunun belirlenmesi ve bu soruna bağlı alt sorunların da tanımlanmasıdır.

Hedef analizi ise, sorun analizinde belirlenen temel sorunu ve alt sorunları olumluya çeviren önermeler ortaya koymaktır.

‘Amaç’ kısmının yazılması

- Bu açıklamalar doğrultusunda, denilebilir ki projenin genel amacı, **temel sorunun** çözümüne yönelik bir **hedef**, özel amaç(lar) ise, alt sorun(lar)un çözümüne yönelik hedef(ler)tir. Özel amaçların, proje sonunda ölçülebilir, ulaşılabilir ve gerçekçi özellikte olması gerekir ve başvuru sahibi, bu amaçların gerçekleştiğini proje sonunda **ispat** etmekle yükümlüdür.

Amaç kısmının yazılması

- Özel amaçlar faaliyetler ile karıştırılmamalıdır. **Örneğin;** Bir eğitimin gerçekleşmesi, bir makinenin alınması özel amaç değil, faaliyettir.
- Eğitim sonucunda katılımcıların teknik kapasite, bilgi ve becerilerinin artırılması, alınan makine ile üretim miktarında artış sağlamak ise faaliyetlerinize bağlı olarak ulaşabileceğiniz özel amaçlardır.

1.1 AMAÇ

Web sunucuları veya aktif ağ cihazları tarafından farklı biçimlerde saklanan web erişim kütüklerinin temizlenmesine yönelik uygun bir yazılım geliştirmektir. Ayrıca, geliştirilen bu yazılım ile temizlenen erişim kütüklerinden anlamlı bilgilerin elde edilmesine yönelik analiz yapılmıştır. Yapılan bu analiz, web sitelerine erişen kullanıcıların davranışları hakkında bilgi edinilmesini ve elde edilen bu bilgilerle kullanıcı davranışlarının tahmin edilmesine olanak sağlamaktadır. Bu geliştirilen uygulamaya ilave olarak ilerleyen aşamalarda diğer saldırı türlerinin tespiti ve web sunucularına entegre olarak anlık tespit yapılması da projenin amaçlarındandır.

Araştırma Projesi

Araştırma projesi deneyimi, öğrenciye araştırma metodolojisini öğrenme ve danışmanıyla yakın bir çalışma olanağı sunar.

Genellikle ileri düzey kavramların, çeşitli deneysel tekniklerin kullanılmasını gerektirir. Literatür taraması tek başına bir araştırma projesi değildir, projede bulunması gereken en önemli unsurlardan biridir.

Araştırma Süreci

Bir konu hakkında araştırma yapılmak istendiğinde öncelikle aşağıda yer alan araştırma süreçlerinden sırasıyla geçilmesi gerekmektedir:

i. Öncelikle bir literatür taraması yapılmalı,
ii. Bu alandaki çalışmalar iyice okunmalı ve notlar çıkarılmalı,
iii. Literatür taraması sonunda araştırma sorusu belirlenmeli,
iv. Analiz yöntemi belirlenmeli,
v. Veriler toplanmalı,
vi. Analiz yapılmalı,
vii. Analiz sonucunda elde edilen bulgular yorumlanmalı,
viii. Son olarak proje raporu yazılmalıdır.

Proje Seçiminde Cevap Aranan Sorular?

- Hangi alanlara katkı sağlayacak
- Ne düzeyde katkı sağlayacak
- Ne kadar değerlidir
- Ne kadar anlamlıdır
- Hangi soruna/sorunlara nasıl çözüm bulmaktadır
- Özgün/yenilikçi yaklaşımı var mıdır?
- Kim/kimler için önemlidir. Hedef kitle!
- Geliştirilebilir yanları var mıdır? vb..

ÖRNEKLERLE PROJE İSİMLENDİRME
AMAÇ VE HEDEF BELİRLEME

Yazılım Mühendisliği

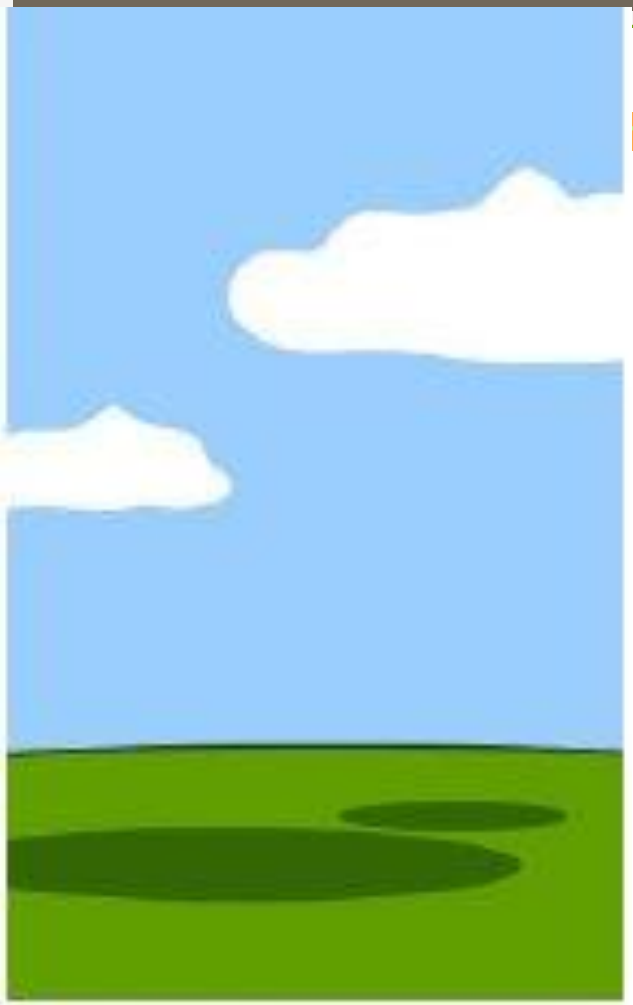
Bölüm - 3

Planlama

Müşterinin İstediği

Analistin Anladığı

Yazılımcının Geliştirdiği



Projenin Dokümantasyonu

Müşterinin Asıl İhtiyacı

Planlama

- Yazılım geliştirme sürecinin ilk aşaması
- Başarılı bir proje geliştirebilmek için projenin tüm resminin çıkarılması işlemi
- Proje planlama aşamasında yapılan işlemler
  - Proje Kaynaklarının Belirlenmesi
  - Proje Maliyetlerinin Kestirilmesi
  - Proje Ekip Yapısının Oluşturulması
  - Ayrıntılı Proje Planının Yapılması
  - Projenin İzlenmesi
- Proje planı tüm proje süresince sürekli olarak kullanılacak, güncellenecek ve gözden geçirilecek bir belgedir

Proje Kaynakları

- İnsan Kaynakları
- Donanım Kaynakları
- Yazılım Kaynakları

Planlama; bu kaynakların tanımını yapar ve

zaman kullanımı,

görev süreleri,

edinilme zamanlarını

planlar

İnsan Kaynakları

- Planlama; hangi tür elemanların, hangi süre ile ve projenin hangi aşamalarında yer alacağını belirler

Proje Yöneticisi	Donanım Ekip Lideri
Yazılım Ekip Lideri	Donanım Mühendisi
Web Tasarımcısı	Ağ Uzmanı
Sistem Tasarımcısı	Yazılım Destek Elemanı
Programcı	Donanım Destek Elemanı
Sistem Yöneticisi	Eğitmen
Veri Tabanı Yöneticisi	Denetleyici
Kalite Sağlama Yöneticisi	Çağrı Merkezi Elemanı

Donanım Kaynakları

- Günümüzde giderek açık sistem mimarisine dönüşmektedir.
- Donanım Kaynakları:
  - Ana Bilgisayarlar
  - Sunucular (Web, E-posta, Veri Tabanı)
  - Kullanıcı Bilgisayarları (PC)
  - Yerel Alan Ağı (LAN) Alt Yapısı
  - Geniş Alan Ağı (WAN) Alt Yapısı
- Yazılımın geliştirileceği ortam, gerçek kullanım ortamı dışında olmalıdır.
- Öte yandan, geliştirme ve uygulama ortamlarının aynı konfigürasyonda olmaları, ileride kurulum sırasında ortaya çıkabilecek taşıma sorunlarını büyük ölçüde giderecektir.

Yazılım Kaynakları

- Büyük ölçekte otomatik hale getirilmiş ve bilgisayar destekli olarak kullanılmaktadır.
- Bilgisayar Destekli Tasarım (CAD) ve Bilgisayar Destekli Mühendislik (CASE) araçları olarak bilinmektedirler.

Yazılım Kaynakları

- İş sistemleri planlama araçları
  - İş akış yapısının üst modelinin üretilmesinde kullanılır.
  - Bilgi akışı, bilgi yapısı iş birimlerindeki tıkanıklıklar bu araçlar kanalıyla ortaya çıkarılır.
- Proje yönetim araçları
  - Yönetici tarafından, projede yapılan işlerin izlenmesi, kaynak ataması, proje iş yapısının üretilmesi, gözlenen değerlerin işlenmesini sağlayan araçlar.
- Analiz ve tasarım araçları
  - Kullanılan modelleme tekniklerini ayrı ayrı ya da bütünleşik olarak uygulayan araçlar. Üretilen modelin kalitesinin ölçülmesi
- Programlama araçları
  - Derleyiciler, nesne-tabanlı programlama araçları, görsel programlama platformları.

Yazılım Kaynakları (devam)

- Test araçları
  - Yazılımı doğrulama ve geçerleme işlemlerinde kullanılır. Test verisi üreticiler, otomatik test yordamları, ...
- Prototipleme ve simülasyon araçları
  - Geliştirmenin erken aşamalarında kullanıcıya, sonuç ürünün çalışması ile ilgili fikir veren ve yönlendiren araçlar.
- Bakım araçları
  - Programın bakımını kolaylaştıran, bir kaynak koddan program şemalarının üretilmesini, veri yapısının ortaya çıkarılmasını sağlayan araçlar.
- Destek araçları
  - İşletim sistemleri, ağ yazılımları, e-posta ve ortam yönetim araçları.

Proje Maliyetleri

- Maliyet kestirimi; bir bilgi sistemi ya da yazılım için gerekebilecek iş gücü ve zaman maliyetlerinin üretimden önce belirlenebilmesi için yapılan işlemlerdir.

- Kullanılan Unsurlar
  - Geçmiş projelere ilişkin bilgiler
  - Proje ekibinin deneyimleri
  - İzlenen geliştirme modeli

  birden çok kez uygulanabilir

Proje Maliyetleri

Maliyet yönetimi sayesinde;

- Gecikmeler önlenir
- Bilgi sistemi geliştirme süreci kolaylaştırılır
- Daha etkin kaynak kullanımı sağlanır
- İş zaman planı etkin olarak gerçekleştirilir
- Ürün sağlıklı olarak fiyatlandırılır
- Ürün zamanında ve hedeflenen bütçe sınırları içerisinde bitirilir

Gözlemlenebilecek değerler

- Projenin toplam süresi
- Projenin toplam maliyeti
- Projede çalışan eleman sayısı, niteliği, çalışma süresi
- Toplam satır sayısı
- Bir satırın maliyeti (ortalama)
- Bir kişi/ay'da gerçekleştirilen satır sayısı
- Toplam işlev sayısı
- Bir işlevin maliyeti
- Bir kişi/ay'da gerçekleştirilen işlev sayısı
- Bir kişi/ay'da maliyeti

Proje Sınıfları

- Ayrık Projeler:
  - Boyutları küçük,
  - Deneyimli personel tarafından gerçekleştirilmiş
  - LAN üzerinde çalışan insan kaynakları yönetim sistemi gibi
- Yarı Gömülü:

  Hem bilgi boyutu hem donanım sürme boyutu olan projeler
- Gömülü Projeler:

  Donanım sürmeyi hedefleyen projeler (pilotsuz uçağı süren yazılım - donanım kısıtları yüksek)

İşlev Noktaları Yönetimi

İşlev Noktaları Yöntemi

İşlev noktaları, geliştirmenin erken aşamalarında (çözümleme aşamasında) saptanan bir değerdir. Bu değer, sistemin oluşturulduğu uygulama geliştirme ortamından bağımsız olarak elde edilmektedir. İşlev noktalarının hesaplamasında problem tanımı girdi olarak alınarak üç temel adım izlenir:

1. Problemin bilgi ortamının incelenmesi
2. Problemin teknik karmaşıklığının incelenmesi
3. İşlev noktası hesaplama

Özellikle, çözümleme çalışması, yapısal yöntemlerle yapıldığında, işlev noktaları yarı otomatik bir biçimde kolayca elde edilebilir. İşlev noktaları yönteminin proje geliştirme sürecinin erken aşamalarında oldukça etkili olduğu belirtilmektedir.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Girdileri

Yazılıma girdi olarak verilen her farklı uygulama bileşeni bir kulanıcı girdisi olarak sayılır. Kulanıcı girdilerinin sayısı hesaplanırken, alan bazında değil, daha genel olarak mantıksal kayıt bazında uygulama yapılmalıdır. Örneğin, bir personel sisteminin geliştirilmesinden söz edildiğinde, personel sicil bilgileri, personel izin bilgileri gibi bilgiler ayrı kullanıcı girdileri olarak sayılabilir. Personel sicil no, personel adı soyadı türündeki alan bilgileri kullanıcı girdisi olarak sayılmamalıdır.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Çıktıları

Kullanıcıyı ilgilendiren her tür mantıksal çıktı, kullanıcı çıktısı olarak sayılmalıdır. Bu kapsamda kullanıcı çıktılarına örnek olarak, raporlar, ekran çıktıları, hata iletileri vb. verilebilir. Girdiler gibi çıktılar da genel olarak sayılmalıdır. Bir raporun içerisindeki bir alan kullanıcı çıktısı olarak sayılmamalıdır. Öte yandan, aynı bilgileri içeren ancak bilgi düzeni ya da sıralaması farklı olan raporlar da ayrı kullanıcı çıktısı olarak ele alınmamalıdır. Örneğin, soyadına göre sıralanmış personel listesi ve sicil no'suna göre sıralanmış personel listesi raporları farklı kullanıcı çıktıları olarak sayılmamalıdır.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Sorguları

Kullanıcı sorgusu, çevrim içi olarak bilgisayara verilen bir girdi sonucu yine çevrim içi olarak bir kullanıcı çıktısı alınması biçiminde tanımlanmaktadır. Her farklı sorgu ayrı olarak sayılmalıdır. Örneğin; personel sicil bilgilerinin sorgulanması, personel maaş bilgilerinin sorgulanması vb.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kütükler

Her mantıksal bilgi yığını ya da kütük ayrı olarak sayılmalıdır. Personel sicil kütüğü, personel maaş kütüğü, personel hareket kütüğü vb.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Dışsal Arayüzler

Geliştirilmesi öngörülen bilgi sisteminin, gerek kurum içinde gerekse kurum dışında bir başka bilgi sistemi ile bilgisayar ortamında bir iletişimi (çevrim dışı disket vb. ya da çevrim-içi doğrudan bağlantı) söz konusu ise bu durum bir dışsal arayüz olarak sayılmalıdır. Kağıt ya da rapor türündeki arayüzler bir dışsal arayüz sayılmazlar. Örneğin, personel bilgi sistemi, yine aynı kurumda olduğu varsayılan muhasebe bilgi sistemine, personel maaşlarına ilişkin bilgiyi bilgisayar ortamında aktarması bir dışsal arayüz olarak sayılabilir.

- Bunların ağırlık faktörleriyle çarpımları toplanarak, Ayarlanmamış İşlev Nokta (AİN) sayısı hesaplanır.

Problem Teknik Karmaşıklığının İncelenmesi - TKF

- Bu adımda, probleme ilişkin dışsal etkenlerin değerlendirilmesi yapılır. Problem **Teknik Karmaşıklık Faktörü (TKF)**, tablo 3.3'de belirtilen ve problemin karmaşıklığına ilişkin sorulara verilecek yanıtların toplamıyla elde edilir. Uygulama ortamının ve uygulamanın özellikleri dikkate alınarak her soruya 0 ile 5 arasında bir yanıt verilir. Örneğin, 8. soruyu dikkate alalım. Eğer uygulama, ana veritabanının, tümüyle çevrim içi olarak güncellenmesini gerektiriyorsa bu durumda yanıtımız 5 olacaktır. Ancak, bazı veri tabanı işlemleri (örneğin %10 gibi) çalışma saatleri dışında, çevrim dışı olarak yapılıyorsa bu durumda 4 gibi bir yanıt verilebilir.

Yanıt Kılavuzu:

0: Hiçbir Etkisi Yok
1: Çok Az etkisi var
2: Etkisi Var
3: Ortalama Etkisi Var
4: Önemli Etkisi Var
5: Mutlaka Olmalı, Kaçınılamaz

1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?
2. Veri iletişimi gerekiyor mu?
3. Dağıtık işlem işlevleri var mı?
4. Performans kritik mi?
5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?
6. Sistem, çevrim içi veri girişi gerektiriyor mu?
7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?
8. Ana kütükler çevrim-içi olarak mı günleniyor?
9. Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı?
10. İçsel işlemler karmaşık mı?
11. Tasarlanacak kod, yeniden kullanılabilir mi olacak?
12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?
13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?
14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?

- Cevaplar 0 ile 5 arasında puanlandırılır
- Bunlar hesaplanıp toplanarak Teknik Karmaşıklık Faktörü (TKF) elde edilir.

İşlev Nokta Sayısı Hesaplama

Bu son adımda, bilinen ve sık kullanılan deneysel formül kanalıyla ilk iki adım sonucundaki bulgular kullanılarak işlev nokta sayısı belirlenir.

Gerek AİN, gerekse TKF önceki bölümlerde açıklandığı gibi hesaplandıktan sonra, geliştirilecek bilgi sistemine ilişkin İşlev Nokta Sayısı (İN) aşağıdaki formül kullanılarak hesaplanır.

İşlev Nokta Sayısı çeşitli amaçlarla kullanılabilir:
İşlev Noktaları, başka yöntemlerle birlikte kullanılmak istenildiğinde, kullanılacak yazılım geliştirme platformuna göre aşağıdaki değerler kullanılarak kolayca satır-sayısı kestirimine dönüştürülebilir.

İN = AİN x (0.65 x 0.01 x TKF)

Üretkenlik = İN / Kişi-Ay
Kalite = Hatalar / İN
Maliyet = $ / FP

İşlev Noktaları, başka yöntemlerle birlikte kullanılmak istenildiğinde, kullanılacak yazılım geliştirme platformuna göre aşağıdaki değerler kullanılarak kolayca satır-sayısı kestirimine dönüştürülebilir.

Örneğin İN kestirimi olarak 300 İN bulduğumuzu varsayalım. Eğer nesne-kökenli bir dil kullanarak (smalltalk gibi) yazılımı geliştirecek isek yaklaşık satır sayısı kestirimi **300x30=9000** satır olacaktır.

Programlama Platformu	Satır Sayısı/İN (Ortalama)
Assembly Dili	300
COBOL	100
FORTRAN	100
Pascal	90
C	90
Ada	70
Nesne-Kökenli Diller	30
4. Kuşak Dileri	20
Kod Üreticiler	15

Etkin Maliyet Modeli - COCOMO

Tüm COCOMO modelleri, temel girdi olarak satır sayısı kestirimini alır ve çıktı olarak iş-gücü (Kişi-ay, Kişi-yıl vb.) ve zaman (ay, yıl, hafta vb.) çıktılarını verir .

- İş Gücü (K) $K=a*S^b$
- Zaman (T) $T=c*K^d$

a,b,c,d : her bir model için farklı katsayılar

S : bin türünden satır sayısı

Proje Sınıfları

- **Ayrık Projeler:**
  - Boyutları küçük,
  - Deneyimli personel tarafından gerçekleştirilmiş
  - LAN üzerinde çalışan insan kaynakları yönetim sistemi gibi
- **Yarı Gömülü:**

  Hem bilgi boyutu hem donanım sürme boyutu olan projeler
- **Gömülü Projeler:**

  Donanım sürmeyi hedefleyen projeler (pilotsuz uçağı süren yazılım - donanım kısıtları yüksek)

Temel Model

- Küçük-orta boy projeler için hızlı kestirim yapmak amacıyla kullanılır
- Dezavantajı: Yazılım projesinin geliştirileceği ortam ve yazılımı geliştirecek ekibin özelliklerini dikkate almaz
- Avantajı: Hesap makinesi ile kolaylıkla uygulanabilir

Temel Model

- Ayrık Projeler
  - İş Gücü $K=2.4*S^{1,05}$
  - Zaman $T=2.5*K^{0,38}$
- Yarı Gömülü Projeler
  - İş Gücü $K=3,0*S^{1,12}$
  - Zaman $T=2.5*K^{0,35}$
- Gömülü Projeler
  - İş Gücü $K=3,6*S^{1,20}$
  - Zaman $T=2.5*K^{0,32}$

Araştırma ödevleri

- Projeniz için gereksinim analizini yazınız
- Süreç modelini belirleyiniz
- Maliyet kestirimini öğrendiğiniz yöntemlere göre hesaplayınız
- Gömülü ve yarı gömülü sistemlere örnekler veriniz

FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
Yazılım Mühendisliği Bölümü

YMH114 - Yazılım Mühendisliğinin Temelleri Dersi
Proje Uygulaması ve Dokümantasyonu

AKILLI ŞEHİR UYGULAMALARININ İNCELENMESİ VE ÖRNEK BİR UYGULAMA

Geliştiren

Ahmet ÇAKMAK

Proje Yürütücüleri
Yrd. Doç. Dr. Murat KARABATAK
Dr. Muhammet BAYKARA

Mayıs – 2017

ÖNSÖZ VE TEŞEKKÜR

Bu proje çalışması süresince her zaman desteklerini hissettiğimiz ve kendilerinden istifade ettiğimiz YMH 114 dersi sorumlu öğretim üyesi Yrd. Doç. Dr. Murat KARABATAK ve Dr. Muhammet BAYKARA hocalarımıza teşekkür ederim.

Ayrıca hayatım boyunca ve bu çalışma süresince desteklerini esirgemeyen ailem ve arkadaşlarıma teşekkürü bir borç bilirim.

Ahmet ÇAKMAK

1. GİRİŞ

1.1 Projenin Amacı

1.2 Projenin Kapsamı

1.3 Tanımlamalar ve Kısaltmalar

2. PROJE PLANI

2.1 Giriş

2.2 Projenin Plan Kapsamı

2.3 Proje Zaman-İş Planı

2.4 Proje Ekip Yapısı

2.5 Önerilen Sistemin Teknik Tanımları

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

2.7 Proje Standartları, Yöntem ve Metodolojiler

2.8 Kalite Sağlama Planı

2.9 Konfigürasyon Yönetim Planı

2.10 Kaynak Yönetim Planı

2.11 Eğitim Planı

2.12 Test Planı

2.13 Bakım Planı

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

3.1.1 Örgüt Yapısı

3.1.2 İşlevsel Model

3.1.3 Veri Modeli

3.1.4 Varolan Yazılım/Donanım Kaynakları

3.1.5 Varolan Sistemin Değerlendirilmesi

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

3.2.2 İşlevsel Model

3.2.3 Genel Bakış

3.2.4 Bilgi Sistemleri/Nesneler

3.2.5 Veri Modeli

3.2.6 Veri Sözlüğü

3.2.7 İşlevlerin Sıradüzeni

3.2.8 Başarım Gerekleri

3.3 Arayüz (Modül) Gerekleri

3.3.1 Yazılım Arayüzü

3.3.2 Kullanıcı Arayüzü

3.3.3 İletişim Arayüzü

3.3.4 Yönetim Arayüzü

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelenmesi

3.4.2 Eğitim Belgeleri

3.4.3 Kullanıcı El Kitapları

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

4.1.2 Varsayımlar ve Kısıtlamalar

4.1.3 Sistem Mimarisi

4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

4.1.4.2 Veri Arabirimleri

4.1.4.3 Diğer Sistemlerle Arabirimler

4.1.5 Veri Modeli

4.1.6 Testler

4.1.7 Performans

4.2 Veri Tasarımı

4.2.1 Tablo tanımları

4.2.2 Tablo- İlişki Şemaları

4.2.3 Veri Tanımları

4.2.4 Değer Kümesi Tanımları

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım

4.3.2 Modüller

4.3.2.1 Yönetici Modülü

4.3.2.1.1 İşlev

4.3.2.1.2 Kullanıcı Arabirimi

4.3.2.1.3 Modül Tanımı

4.3.2.1.4 Modül iç Tasarımı

4.3.2.2 Seçmen Modülü

4.3.2.2.1 İşlev

4.3.2.2.2 Kullanıcı Arabirimi

4.3.2.2.3 Modül Tanımı

4.3.2.2.4 Modül iç Tasarımı

4.3.3 Kullanıcı Profilleri

4.3.4 Entegrasyon ve Test Gereksinimleri

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.1 Ortak Alt Sistemler

4.4.2 Modüller arası Ortak Veriler

4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

4.4.4 Güvenlik Altsistemi

4.4.5 Veri Dağıtım Altsistemi

4.4.6 Yedekleme ve Arşivleme İşlemleri

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1. Giriş

5.2. Yazılım Geliştirme Ortamları

5.2.1 Programlama Dilleri

5.2.2 Veri Tabanı Yönetim Sistemleri

5.2.2.1 VTYS Kullanımının Ek Yararları

5.2.2.2 Veri Modelleri

5.2.2.3 Şemalar

5.2.2.4 VTYS Mimarisi

5.2.2.5 Veritabanı Dilleri ve Arabirimleri

5.2.2.6 Veri Tabanı Sistem Ortamı

5.2.2.7 VTYS'nin Sınıflandırılması

5.2.2.8 Hazır Program Kütüphane Dosyaları

5.2.2.9 CASE Araç ve Ortamları

5.3. Kodlama Stili

5.3.1 Açıklama Satırları

5.3.2 Kod Biçimlemesi

5.3.3 Anlamlı İsimlendirme

5.3.4 Yapısal Programlama Yapıları

5.4. Program Karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

5.5. Olağan Dışı Durum Çözümleme

5.5.1 Olağandışı Durum Tanımları

5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

5.6. Kod Gözden Geçirme

5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

5.6.2.1 Öbek Arayüzü

5.6.2.2 Giriş Açıklamaları

5.6.2.3 Veri Kullanımı

5.6.2.4 Öbeğin Düzenlenişi

5.6.2.5 Sunuş

6. DOĞRULAMA VE GEÇERLEME

6.1. Giriş

6.2. Sınama Kavramları

6.3. Doğrulama ve Geçerleme Yaşam Döngüsü

6.4. Sınama Yöntemleri

6.4.1 Beyaz Kutu Sınaması

6.4.2 Temel Yollar Sınaması

6.5.Sınama ve Bütünleştirme Stratejileri

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme
6.6. Sınama Planlaması
6.7. Sınama Belirtimleri
6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri
7. BAKIM
7.1 Giriş
7.2 Kurulum
7.3 Yerinde Destek Organizasyonu
7.4 Yazılım Bakımı
7.4.1 Tanım
7.4.2 Bakım Süreç Modeli
8. SONUÇ
9. KAYNAKLAR

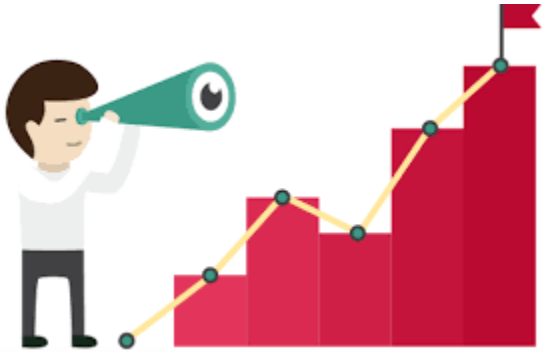
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

6.Hafta

TASARIM

Bölüm Hedefi

- Bu bölümde, yapısal sistem geliştirme metodolojisine göre yapılan bir çözümlemenin tasarıma dönüştürülmesi, tasarım kavramları, veri tasarım ve süreç tasarım yöntemleri açıklanmaktadır. Her tür yazılım için gerekecek ortak alt sistemler tanıtılmaktadır.



Tasarıma Giriş

- Tasarım, Sistem Analizi çalışması sonucunda üretilen Mantıksal Modelin Fiziksel Modele dönüştürülme çalışmasıdır.

- Fiziksel Model geliştirilecek yazılımın;
  - hangi parçalardan oluşacağını,
  - bu parçalar arasındaki ilişkilerin neler olacağını,
  - parçaların iç yapısının ayrıntılarını,
  - gerekecek veri yapısının fiziksel biçimini (dosya, veri tabanı, hash tablosu, vektör, vs)

  tasarımını içerir.

Çözümleme-Tasarım İlişkisi

Kuramsal

Teknik

SRS→SAD

Tasarım Kavramları

- Soyutlama (abstraction): Detayları gizleyerek yukarıdan bakabilme imkanı sağlanır.

- İyileştirme (enhancement): Soyutlama düzeyinde irdeleme bittikten sonra, daha alt seviyelere inilerek tanımlamalarda ayrıntı, bazen de düzeltme yapılarak tasarımın daha kesinlik kazanması sağlanır.

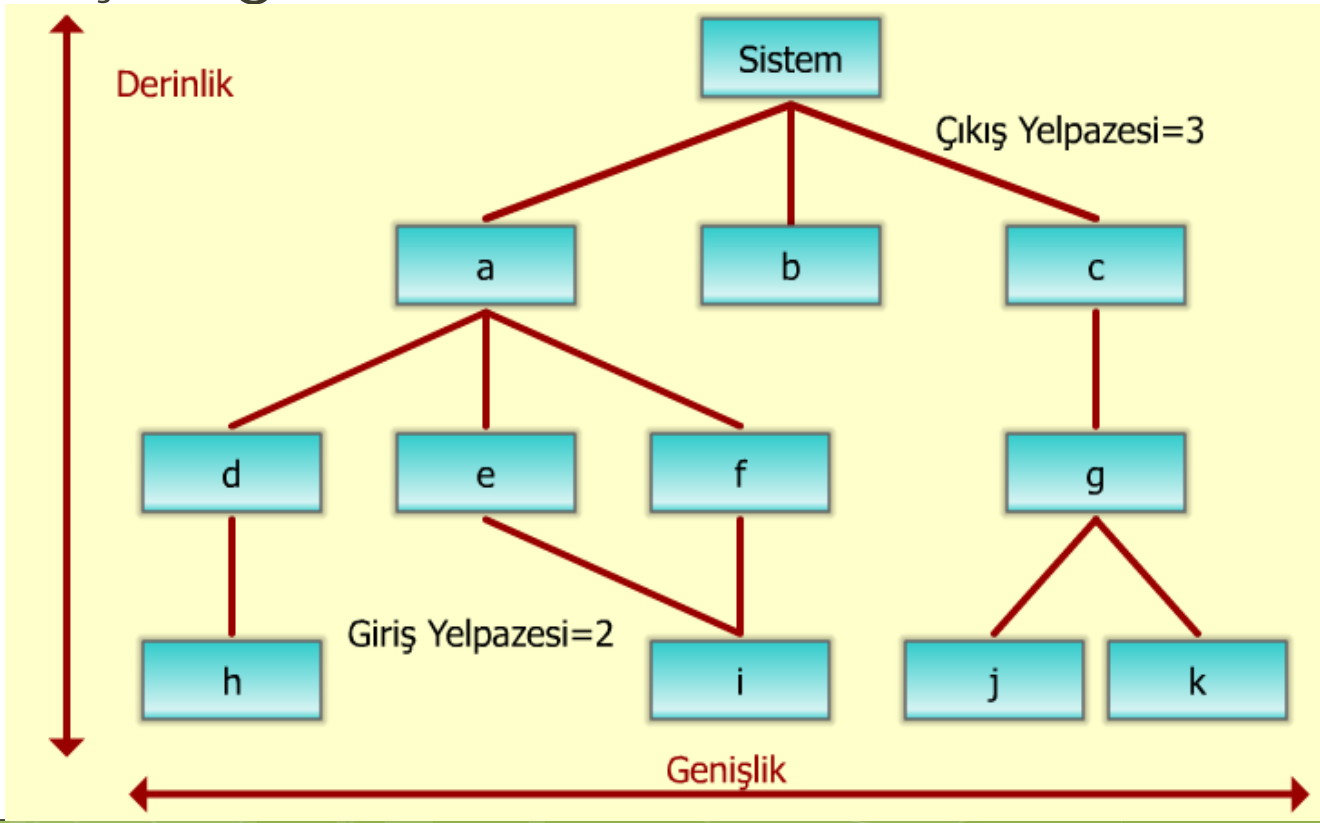
- Modülerlik (modularity): Sistemi istenen kalite faktörleri ışığında parçalara ayrıştırma sonucu elde edilir.

Modülerlik

- Bütün karmaşıklığın tek bir modülde toplanması yerine, anlaşılabilir ve dolayısıyla projenin zihinsel kontrol altında tutulması için sistem bir çok modüle ayrılır.

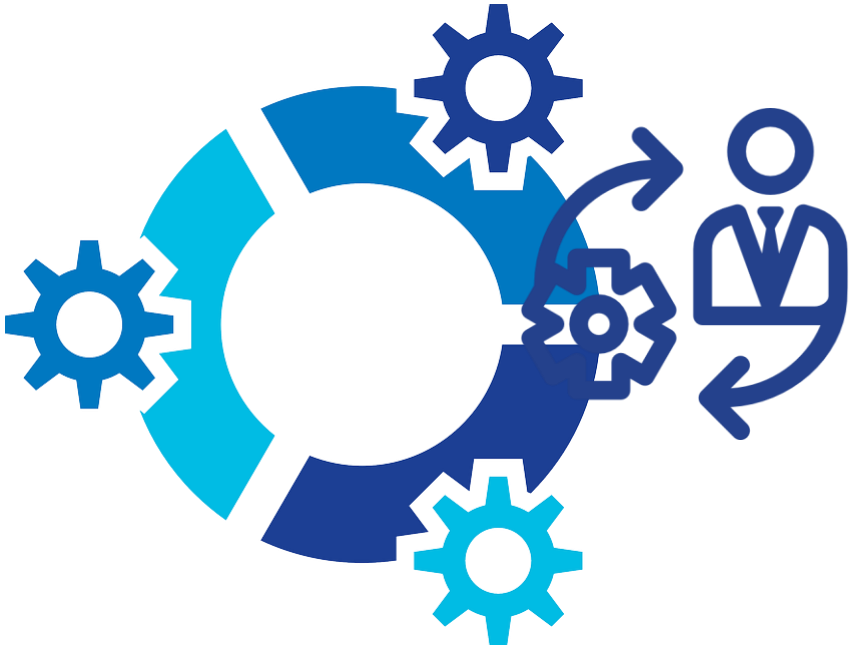
- Modüller, isimleri olan tanımlanmış işlevleri bulunan ve hedef sistemi gerçekleştirmek üzere tümleştirilen birimlerdir.

- Şekil sistemin modüllerden oluşmasını simgelemektedir. Aynı zamanda, sistemin '**kontrol hiyerarşisi**'ni de belirtmektedir. Bir modülün hangi modülleri kullanarak işlevini gerçekleştirdiği bu hiyerarşide görünmektedir.



İşlevsel Bağımsızlık

- Modüllere parametre ile veri gönderilir ve sonuç değer alınır. Bu modülü çağıran program parçası sadece bu sonucu kullanabilir. Çağrılan modülün işlevsel olarak yaptıkları ile ilgili değildir.



Veri Tasarımı

- Yapı Tasarımı, arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.
- Bilgi saklama ve soyutlama bu işlem için önemli kavramlardır.

Veri tasarımı yaparken dikkat edilecek konular

- Değişik veri yapıları değerlendirilmelidir.
- Bütün veri yapıları ve bunlar üzerinde yapılacak işlemler tanımlanmalıdır.
- Alt düzeyde tasarım kararları tasarım süreci içerisinde geciktirilmelidir.
- Bazı çok kullanılan veri yapıları için bir kütüphane oluşturulmalıdır.
- Kullanılacak programlama dili soyut veri tiplerini desteklemelidir.

Yapısal Tasarım

- Yapısal Tasarımın ana hedefi modüler bir yapı geliştirip modüller arasındaki kontrol ilişkilerini temsil etmektir.
- Ayrıca yapısal tasarım bazen de veri akışlarını gösteren biçime dönüştürülebilir.
- Veri Akışları Üç parçada incelenebilir
  - Girdi Akışı
  - Çıktı Akışı
  - İşlem Akışı

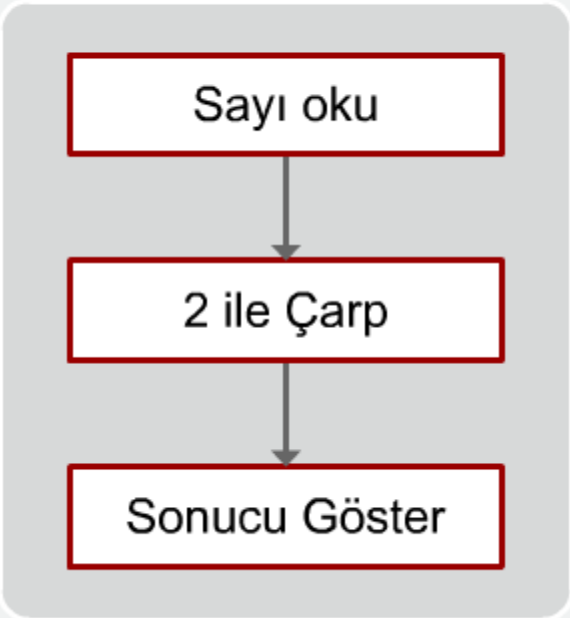
Ayrıntı Tasarım- Süreç Tasarımı

- **Süreç tasarımı;** veri, yapı ve arayüz tasarımından sonra yapılır.
- İdeal şartlarda bütün algoritmik detayın belirtilmesi amaçlanır.
- Ayrıca süreç belirtiminin tek anlamı olması gerekir, değişik şahıslar tarafından farklı yorumlanmamalıdır.
- Doğal diller kullanılabilir (açıklamalarda, çünkü doğal dil tek anlamlı değildir)
- Süreç Tanımlama Dili (PDL)

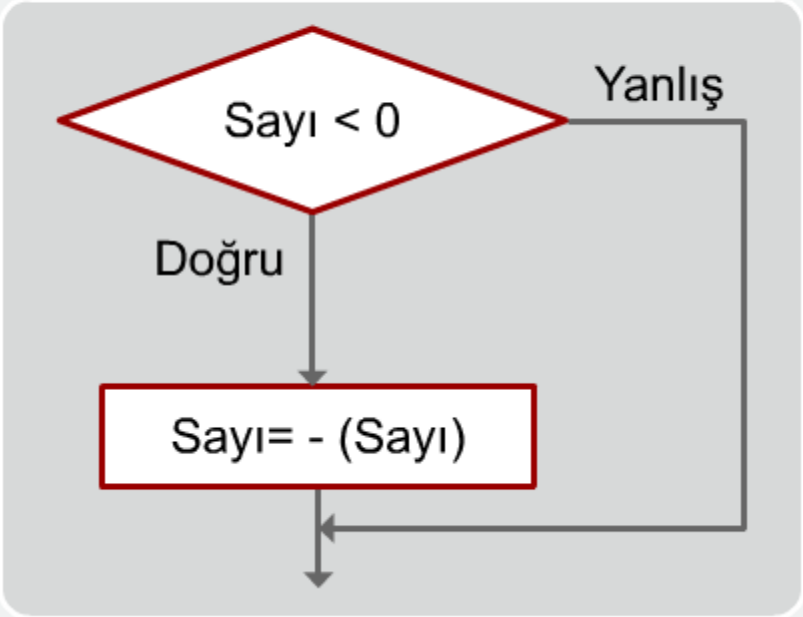
Yapısal Program Yapıları

- Yapısal programlamanın temel amacı;
  - program karmaşıklığını en aza indirmek,
  - program anlaşılabilirliğini artırmaktır.
- Bu amaçla şu yapıları kullanılır;
  - Ardışıl İşlem yapısı
  - Koşullu işlem yapısı
  - Döngü yapısı
- GOTO kullanımı uygun değildir.

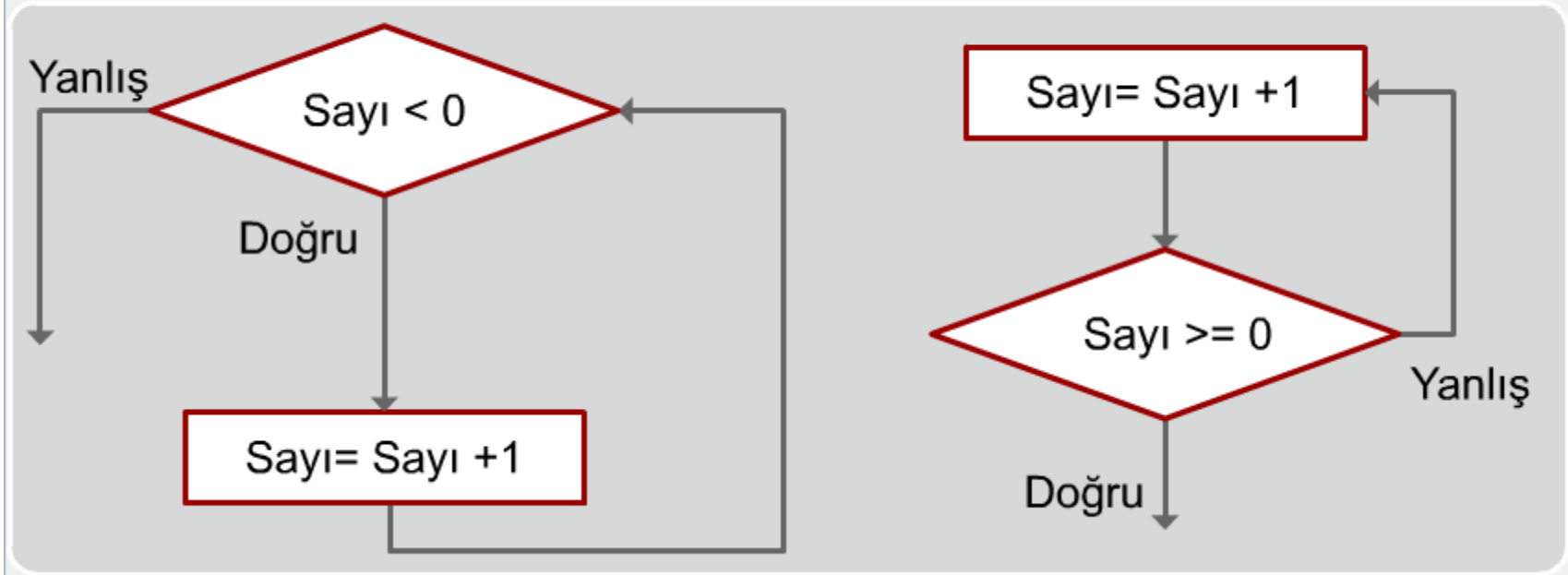
Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Karar Tabloları

- Bazen karmaşık koşul değerlendirmeleri yapmak gerekir. Bunların düzenli bir gösterilimi karar tablolarında yapılabilir.
- Öncelikle, bütün işlemler saptanmalı, sonra ön koşullar belirlenmelidir.
- Belirli işlemler ile belirli koşulları birleştirerek tablo oluşturulur.
- Alt tarafta ise işlemler benzer satırlar olarak gösterilir.

Karar Tablosu Örneği

Kurallar	1	2	3	4
Hesap Geçerli	X	X	X	X
Şifre Geçerli	X		X	X
Yeterli Nakit Var	X			
Yeterli Bakiye Var	X		X	
Bakiye Bildirme İşlemi				X
Para Çekme İşlemi	X			
Para Yatırma İşlemi		X		
Para Gönderme İşlemi			X	

Program Tanımlama Dili

- **Program Tanımlama Dilleri** süreç belirtiminde doğal dillerin programlama dili ile sentezlenmesi şeklinde ortaya çıkmıştır.

- Hangi programlama dilinin kullanılacağından bağımsız özellikler bulunmalıdır.

```
DO
  Hesap Numarasını Oku
  IF (hesap numarası geçerli değil) başlangıca dön
     işlem türünü iste
  IF (para yatırma islemi) { para_yatir(); Başlangıca dön}
  IF (yeterli bakiye yok) başlangıca dön
WHILE
```

Tasarlanması Gereken Ortak Alt Sistemler

- Yetkilendirme altsistemi
- Güvenlik altsistemi
- Yedekleme altsistemi
- Veri transferi altsistemi
- Arşiv altsistemi
- Dönüştürme altsistemi

Yetkilendirme Alt Sistemi

- Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.
  - İşlev bazında yetkilendirme
  - Ekran bazında yetkilendirme
  - Ekran alanları bazında yetkilendirme
- Oracle veri tabanına erişim konusunda yetkilendirme yapmaktadır.

Güvenlik Alt Sistemi

- Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması işlemleri.

- LOG files (Sistem günlüğü)

Yedekleme Alt Sistemi

- Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları veri tabanı (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir.

Veri İletişim Alt Sistemi

- Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleri
- Çevirim içi veri iletimi (real-time)
- Çevirim dışı veri iletimi (disketler, teypler)

Arşiv Alt Sistemi

- Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemlerdir.

- Aktif veri tabanı.

Dönüştürme Alt Sistemi

- Geliştirilen bilgi sisteminin uygulamaya alınmadan önce veri dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması) işlemlerine ihtiyaç vardır.

Kullanıcı Arayüz Tasarımı

- Kullanıcı ile ilişkisi olmayan arayüzler
  - Modüller arası arayüz
  - Sistem ile dış nesneler arası arayüz

- Kullanıcı arayüzleri
  - Kullanım kolaylığı ve öğrenim zamanı esastır
  - Program=arayüz yaklaşımı vardır

Genel Prensipler

- Veri giriş formlarının tutarlı olması
- Önemli silmelerde teyit alınmalı
- Yapılan çoğu işlem geri alınabilmeli
- Hataların affedilmesi, yanlış girişte kırılmama
- Komut isimlerinin kısa ve basit olması
- Menülerin ve diğer etkileşimli araçların standart yapıda kullanımı

Bilgi Gösterimi

- Yalnızca içinde bulunulan konu çerçevesi ile ilgili bilgi gösterilmeli
- Veri çokluğu ile kullanıcı bunaltılmamalı, grafik ve resimler kullanılmalı
- Tutarlı başlık, renkleme ve kısaltma kullanılmalı
- Hata mesajları açıklayıcı ve anlaşılır olmalı
- Değişik tür bilgiler kendi içinde sınıflandırılmalı
- Rakamsal ifadelerde analog görüntü verilmeli (%89 değil)

Veri Girişi

- Kullanıcı hareketleri en aza indirilmeli
- Gösterim ve girdi sahaları birbirinden ayrılmalı (renk)
- Kullanıcı uyarlamasına izin verilmeli, kullanıcı bazı özellikleri tanımlayabilmeli
- Kullanılan konu ile ilgili gereksiz komutlar deaktifleştirilmeli
- Bütün girdiler için yardım kolaylıkları olmalı

Kullanıcı Arayüz Prototipi

- Tasarım çalışması sonucunda, daha önceden gereksinim çalışması sırasında hazırlanmış olan kullanıcı arayüz prototipi, ekran ve rapor tasarımları biçimine dönüşür. Ekranlar son halini alır, raporlar kesinleşir. Kullanıcıya gösterilerek onay alınır.

- Tüm programın tek elden çıktığının ifade edilebilmesi açısından tüm ekranların aynı şablon üzerine oturtulması önerilmektedir.
  - **Menü Çubuğu**
  - **Araç Çubuğu**
  - **Gövde (Değişebilir)**
  - **Durum Çubuğu**

Başlangıç Tasarım Gözden Geçirme

- Yapılan tasarım çalışmasının bir önceki geliştirme aşaması olan analiz aşamasında belirlenen gereksinimleri karşılayıp karşılamadığının belirlenmesidir.
  - Sistem gereksinimlerine yardımcı olan kullanıcılar
  - Sistem analizini yapan çözümleyiciler
  - Sistemin kullanıcıları
  - Tasarımcılar
  - Yönlendirici
  - Sekreter
  - Sistemi geliştirecek programcılar

  dan oluşan bir grup tarafından yapılır.

Ayrıntılı Tasarım Gözden Geçirme

- Başlangıç tasarımı gözden geçirme çalışmasının başarılı bir biçimde tamamlanmasından sonra, tasarımın teknik uygunluğunu belirlemek için Ayrıntılı Tasarım Gözden Geçirme çalışması yapılır. Bu çalışmada;
  - Çözümleyiciler
  - Sistem Tasarımcıları
  - Sistem Geliştiriciler
  - Sekreter

  den oluşan bir ekip kullanılır.

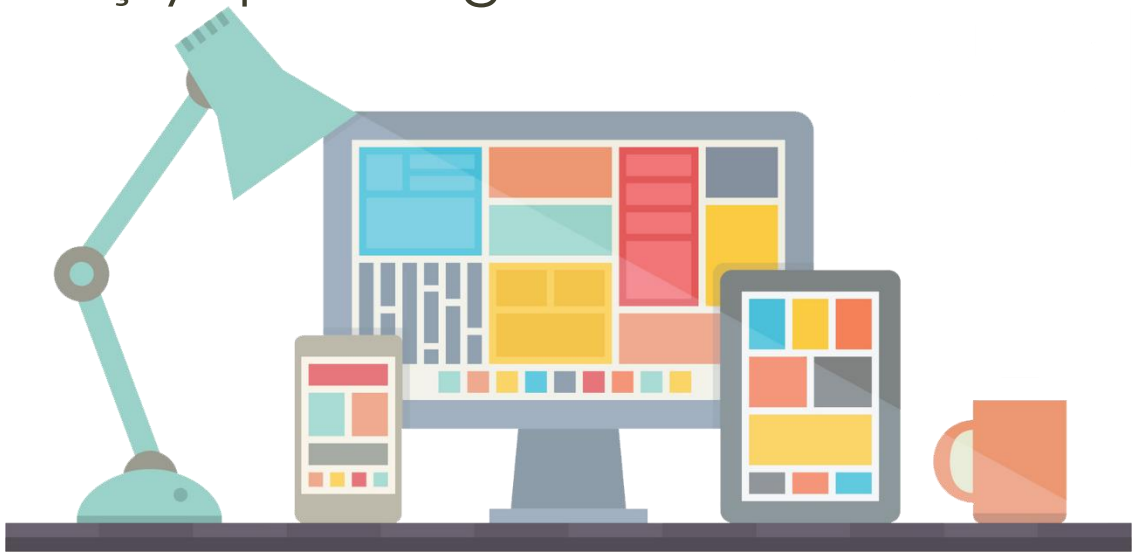
Tasarım Kalite Ölçütleri

- Bağlaşım (Coupling)

  Tasarımı oluşturan modüller arası ilişki ile ilgilidir.

- Yapışıklık (Cohesion)

  Modüllerin iç yapısı ile ilgilidir.



Bağlaşım

- Modüller arası bağlılığın ölçülmesi için kullanılan bir ölçüttür.
- Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır.
- Bağlaşımın düşük olması
  - Hatanın dalgasal yayılma özelliğinin azaltılması
  - Modüllerin bakım kolaylığı
  - Modüller arası ilişkilerde karmaşıklığın azaltılması

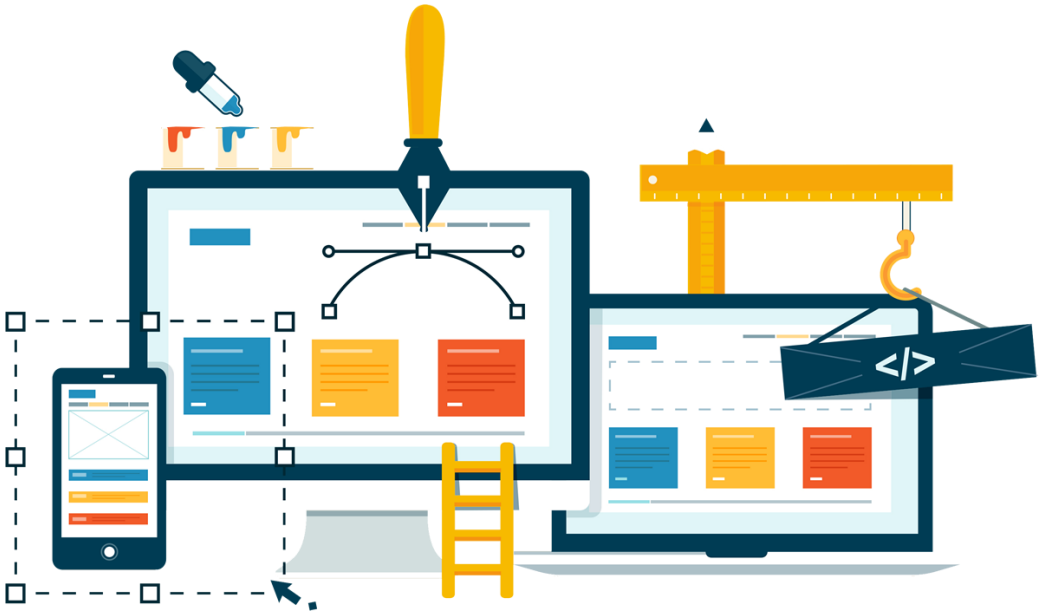
  nedenleri ile istenmektedir

Yalın Veri Bağlaşımı

- Herhangi iki modül arası iletişim yalın veriler (tamsayı, karakter, boolean, vs) aracılığı ile gerçekleştiriliyorsa bu iki modül yalın veri bağlaşımlıdır şeklinde tanımlanır.

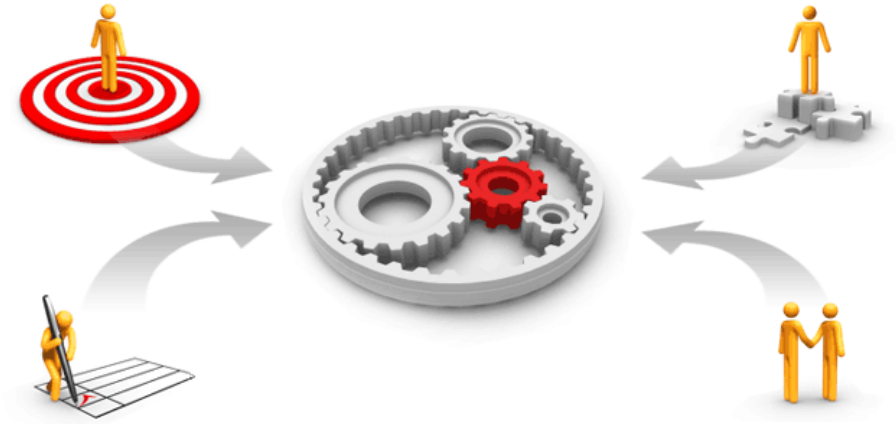
Karmaşık Veri Bağlaşımı

- Herhangi iki modül arasındaki iletişimde kullanılan parametrelerin karmaşık veri yapısı (kayıt, dizi, nesne, vs) olması durumunda modüller karmaşık veri paylaşımlı olarak tanımlanır.



Denetim Bağlaşımı

- İki Modül arasında iletişim parametresi olarak denetim verisi kullanılıyorsa bu iki modül denetim bağlaşımlı olarak tanımlanır.



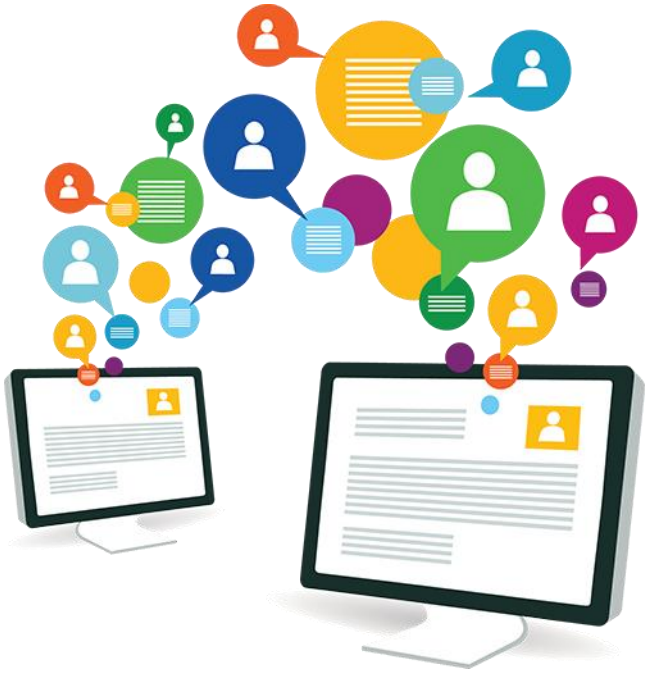
Ortak Veri Bağlaşımı

- Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül ortak veri bağlaşımlı olarak tanımlanır.

- Verilerin ortak veri bağlaşımlı olmaları şu nedenlerden dolayı fazla istenmez;
  - Ortak veri alanını izlemek zordur.
  - Ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler.
  - Ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanacak bütün modüller göz önüne alınmalıdır.

İçerik Bağlaşımı

- Modüllerin iç içe tasarlanması sonucu, bir modülün başka bir modül içerisinde tanımlanmış veri alanına erişebilmesi olanaklaşır ve bu durum içerik bağlaşımına yol açar.



Yapışıklık

- Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir ölçüttür. Modül gücü olarak ta tanımlanır.
- Tasarımda yapışıklık özelliğinin yüksek olması tercih edilir.
- Yapışıklık ile Bağlaşım ters orantılıdır.

İşlevsel Yapışıklık

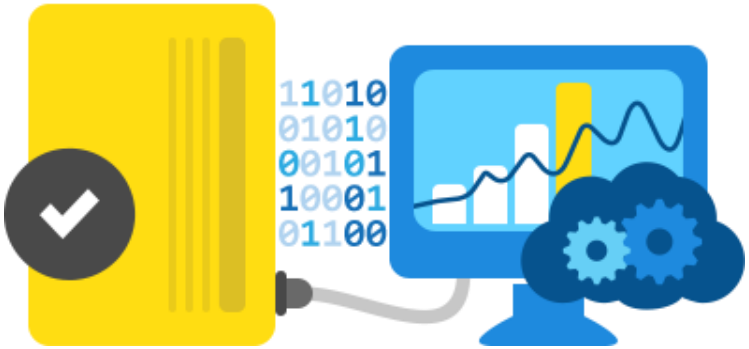
- İşlevsel Yapışık bir modül, tek bir iş problemine ilişkin sorunu çözen modül olarak tanımlanır.

- Maas_Hesapla, Alan_Hesapla gibi

Sırasal Yapışıklık

- Bir modülün içindeki işlemler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanılıyorsa bu modül sırasal yapışık bir modül olarak adlandırılır.

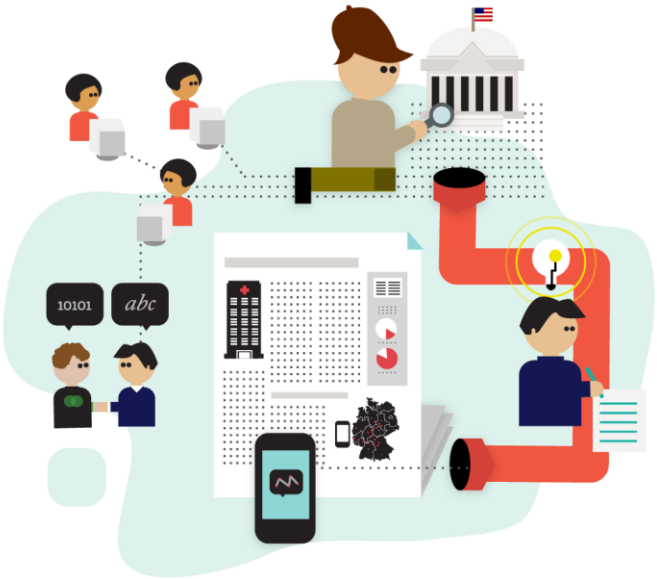
Ham_Veri_Kaydını_Düzelt
Duzeltilmis_Ham_Veri_Kaydini_Dogrula
Dogrulanmis_Kaydi_Gonder



İletişimsel Yapışıklık

- Bir modülün içindeki farklı işlemler aynı girdi ya da çıktıyı kullanıyorlarsa bu modül iletişimsel yapışık bir modül olarak adlandırılır.

Sicil_No_yu_Al
Adres_Bilgisini_Bul
Telefon_Bilgisini_Bul
Maas_Bilgisini_Bul



Yordamsal Yapışıklık

- Yordamsal Yapışık modüldeki işlemler arasında denetim ilişkisi bulunmaktadır.
- İşlemlerin birbirleri ile veri ilişkisi yoktur, ancak işlem sırası önemlidir.

Ekran_Goruntusunu_Yaz

Giris_Kaydini_Oku

Zamansal Yapışıklık

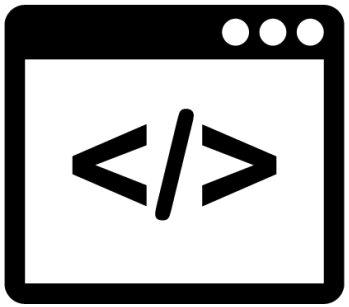
- Bir modül içindeki işlemlerin belirli bir zamanda uygulanması gerekiyor ve bu işlemlerin kendi aralarında herhangi bir ilişkisi yok, yani işlemlerin sırası önemli değil ise, zamansal yapışıklık vardır.

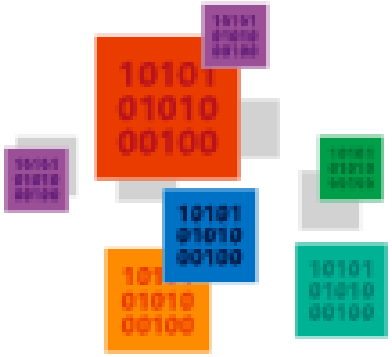
Alarm_Zilini_Ac
Kapiyi_Ac
Kamerayi_Calistir

Mantıksal Yapışıklık

Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller mantıksal yapışık olarak adlandırılır.

Dizilere değer atama işlemleri

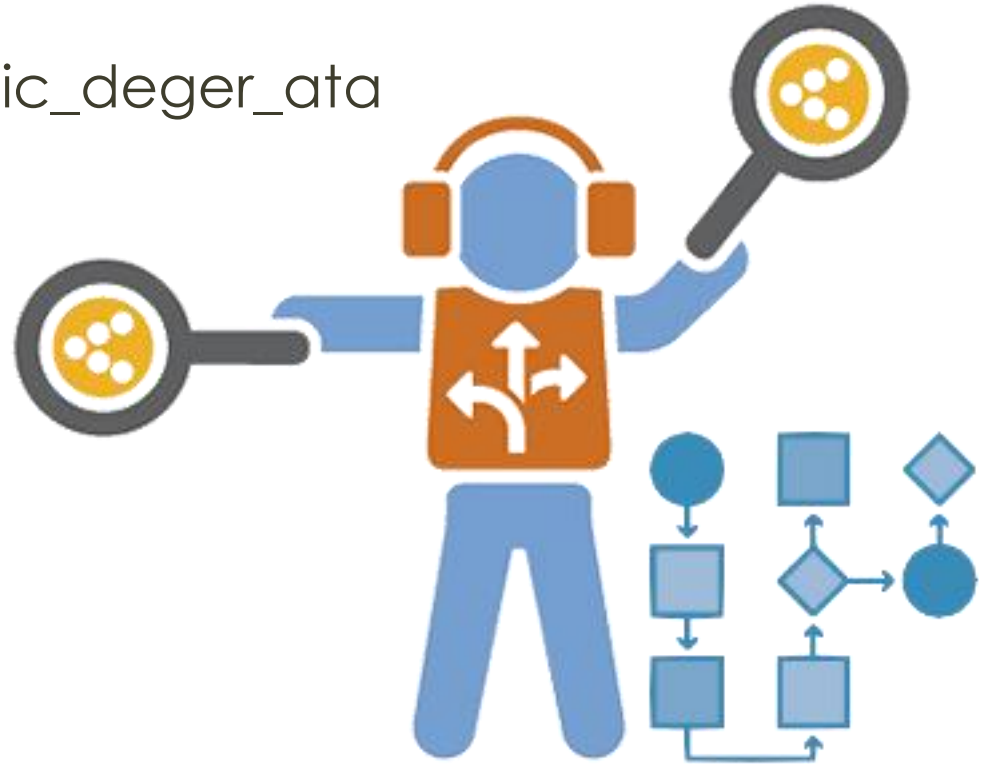




Gelişigüzel Yapışıklık

- İşlemler arasında herhangi bir ilişki bulunmaz.

Ara_Kayit_Oku
B_dizisine_baslangic_deger_ata
Stok_kutugu_oku
Hata_iletisi_yaz

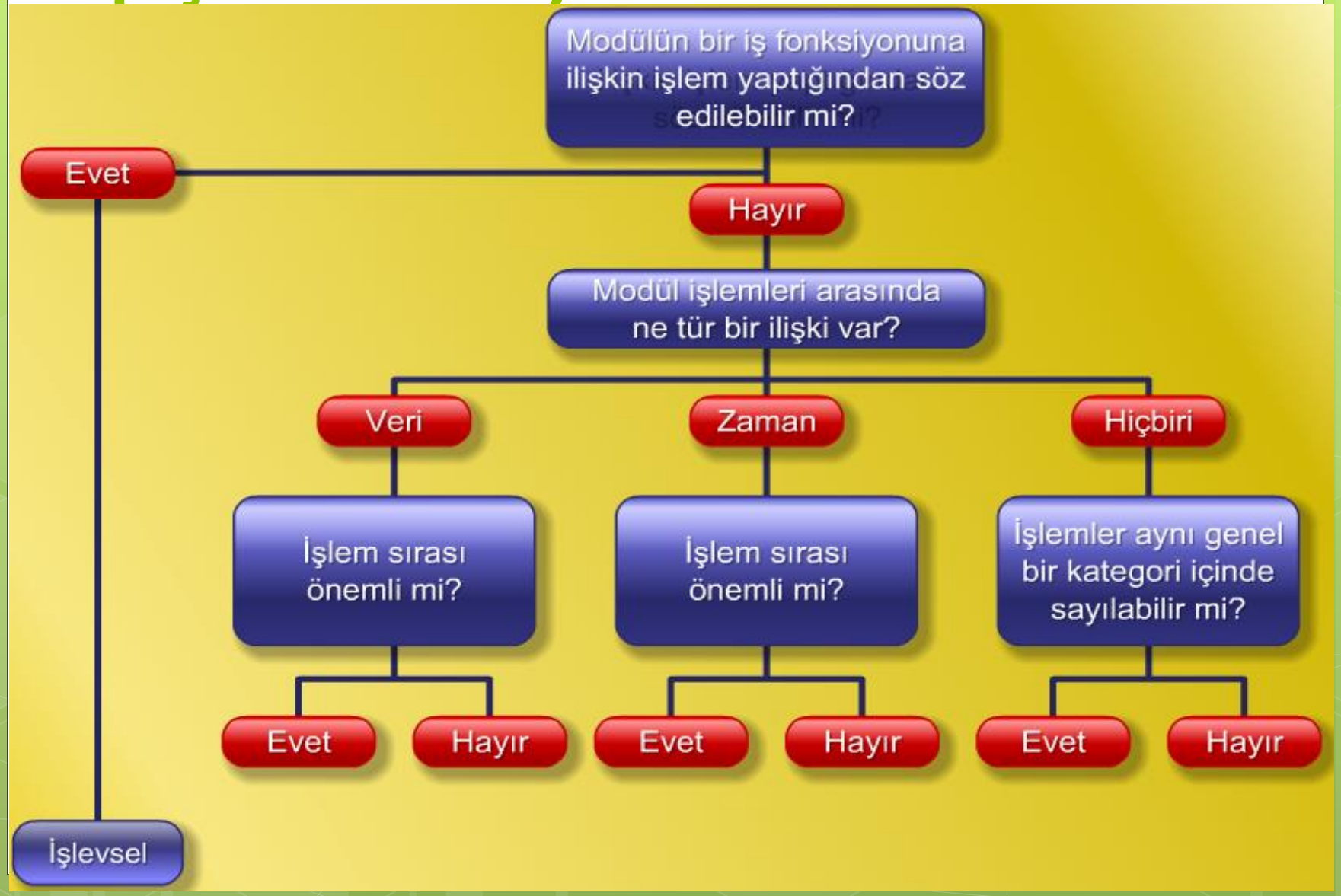


Yapışıklık Hesaplama

- Genel olarak bir modülün yapışıklık düzeyini hesaplamada bazı güçlüklerle karşılaşılabilir. Çünkü, bir modüldeki işlemler hep aynı tür özellik göstermeyebilir.

Yapışıklık Türü	Ölçek	Kalite
İşlevsel	1	En iyi
Sırasal	2	
İletişimsel	3	
Yordamsal	4	
Zamansal	5	
Mantıksal	6	
Gelişigüzel	7	En kötü

Yapışıklık Düzeyine Karar Verilmesi

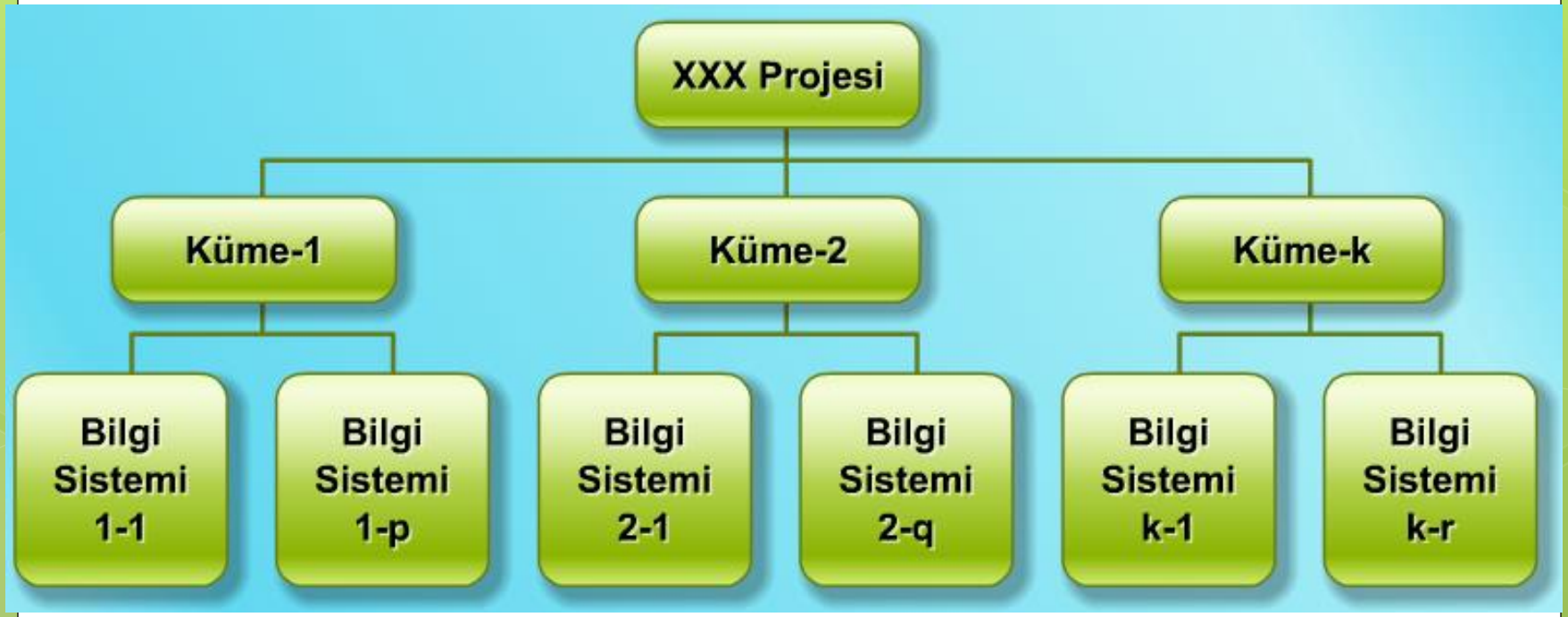


TASARIM RAPORU İÇERİĞİ

- İçeriğin hazırlanmasında IEEE/ANSI standartları kullanılmıştır. Büyük boyutlu projeler için burada verilen içerik, genel mantık aynı kalmak koşulu ile detaylandırılabilir. Örneğin 200 kişi-yıllık işgücü gerektiren bir BT projesi, Şekil'deki gibi bölümlenebilir.

- Ödev IEEE 1016-1998

Tasarım Raporu İçeriği



Örnek Tasarım Raporu İçeriği

1. Giriş

1.1. Kapsam
1.2. Tanımlar, Kısa Adlar ve Kısaltmalar
1.3. Referanslar
1.4. Belgenin Yapısı

2. Genel Tasarım Bilgileri

2.1. Genel Sistem Tanımı
2.2. Varsayımlar ve Kısıtlamalar
2.3. Sistem Mimarisi
2.4. Dış Arabirimler

2.4.1. Kullanıcı Arabirimleri
2.4.2. Veri Arabirimleri
2.4.3. Diğer Sistemlerle Arabirimler

2.5. Veri Modeli
2.6. Testler
2.7. Performans

Örnek Tasarım Raporu İçeriği

3. Veri Tasarımı

3.1. Tablo tanımları
3.2. Tablo- İlişki Şemaları
3.2. Veri Tanımları
3.3. Değer Kümesi Tanımları

4. Süreç Tasarımı

4.1. Genel Tasarım
4.2. Modüller

4.2.1. XXX Modülü

4.2.1.1. İşlev
4.2.1.2. Kullanıcı Arabirimi
4.2.1.3. Modül Tanımı
4.2.1.4. Modül iç Tasarımı

4.2.2. YYY Modülü

4.3. Kullanıcı Profilleri
4.4. Entegrasyon ve Test Gereksinimleri

Örnek Tasarım Raporu İçeriği

5. Ortak Alt Sistemlerin Tasarımı

5.1. Ortak Alt Sistemler
5.2. Modüller arası Ortak Veriler
5.3. Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri
5.4. Güvenlik Altsistemi
5.5. Veri Dağıtım Altsistemi
5.6. Yedekleme ve Arşivleme İşlemleri

EKLER

EK-A Rapor Görünümleri
EK-B Test Belirtimleri
EK-C Kullanıcı Kılavuzu Örneği

Alınan Dersler 1

- Ortak alt sistemler oldukça önemlidir. Tasarlanması ve geliştirilmesi için ayrı bir yazılım ekibi kullanılması gerekir. Aksi durumda, tutarsızlık ve işlem tekrarı gündeme gelir, işgücü kaybına neden olunur.

Alınan Dersler 2

- Özellikle, coğrafik olarak birden çok alana yayılmış kuruluşlar için geliştirilecek bilgi sistemlerinde gerek merkez gerekse bağlı birimlerde çalışacak yazılımın aynı yazılım olarak tasarlanıp geliştirilmesi uygulama ve bakım kolaylığı açısından önerilmektedir.

Alınan Dersler

- Genellikle bu tür uygulamalarda, merkezdeki yazılım, birimlerdeki yazılımlara oranla daha farklı unsurlar içermekte, ancak birimlerdeki uygulamaları da kapsamaktadır.
- Bu durumda, neden merkezdeki yazılımın işlevlerini, kullanılmayacaklarını bile bile birimlere de taşıyalım sorusu akla gelebilir.

Alınan Dersler

- Zira bu durum kaynak kullanımı açısından birimlere ek yük getirmektedir. Bu durumda bakım ve kurulum kolaylığı ve fazla kaynak kullanımı arasında bir tercih yapmak gerekmektedir.
- Günümüzde, bilgisayar kaynaklarının giderek ucuzladığı düşünüldüğünde bu tercihin, bakım ve kurulum kolaylığı yönünde yapılması önerilmektedir.

Alınan Dersler 3

- Genellikle kurumsal BT projelerinde veri iletişiminin tüm türlerine (Çevrim-içi anında, çevrim-içi kargo, çevrim dışı) rastlanmaktadır. Bu durumda, Veri İletişim alt sistemlerinin bu özellikleri içerecek biçimde tasarlanması gerekmektedir.
- Özellikle çevrim içi ve anında iletişim gerektiren durumlarda, işlemlerin izlenmesi ve herhangi bir aksaklık durumunda geri alınması gibi işlevlerin tasarımda içerilmesi gerekmektedir.

Alınan Dersler 4

- Dönüştürme işlemleri, yazılım gereksinim tanımlarında oldukça az ve yetersiz biçimde tanımlanmakta ve bu durum yüklenici ile iş sahibi arasında sorunlara yol açmaktadır.
- Geçmiş uygulamalarda, bilgisayar ortamında saklanan bilgilerin yeni uygulama ortamında oluşturulacak veri tabanına aktarılmasında ortaya çıkan temel sorun, geçmiş verilerdeki tutarsızlıklardır.

Alınan Dersler

- Bir taraftan bu tür tutarsızlıkların yeni sistemde bulunmaması öngörülürken, öte yandan tutarsızlık içeren geçmiş verilerin aktarılması istenir. Bu durum tasarımda içerilmemişse büyük zorluklarla karşılaşılır.
- Örneğin, nüfus uygulamasındaki kişilerin evlilikleri ile ilgili bağ alanlarının dolu olması bir zorunluluk olarak öngörüldüğünü varsayalım.

Alınan Dersler

- Öte yandan, bilgisayar ortamındaki mevcut verilere bakıldığında bir çok bağ alanının boş olduğu görüldüğünde mevcut verilerin aktarımında zorluklar çıkar.
- Veri tabanındaki kısıtlar kaldırılıp veriler olduğu gibi aktarılsa bile, bu verilere ilişkin ileride ortaya çıkabilecek işlemlerde yine sorunlar oluşabilir.

Alınan Dersler

- Bu tür durumların, yazılım gereksinim tanımlarında ayrıntılı olarak tanımlanması, tasarım sırasında dikkatlice ele alınmalarını sağlar.

Sorular

1. Yazılım tasarım sürecinin temel işlemlerini sayınız. E-R diyagramı çizerek ilişkilerini gösteriniz.

2. Geliştireceğiniz bir uygulama için ekran şablonunuzu belirleyiniz.

3. Bağlaşım ve yapışıklık kavramlarını açıklayınız. Program bakımı ile ilişkilerini belirtiniz.

4. Bir sistem tümüyle bağlaşımsız biçimde tasarlanabilir mi? Yani sistemin tüm modülleri arasında hiç bağlaşım olmadan tasarım yapılabilir mi?

5. Tümüyle işlevsel yapışık modüllerden oluşan bir sistem tasarlanabilir mi? Neden yapılabilir? Neden yapılamaz?

6. Bağlaşım ile Yazılım Taşınabilriliği arasındaki ilişkiyi belirtiniz.

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

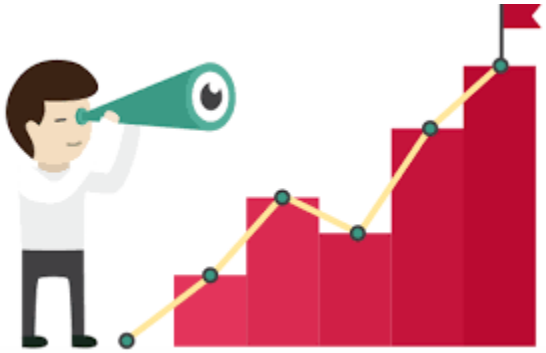
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

7.Hafta

GERÇEKLEŞTİRME

Bölüm Hedefi

- Bu bölümde, yazılım geliştirme ortamları tanıtılmakta ve kodlama yöntemleri açıklanmaktadır. Program karmaşıklığının ölçümü üzerine yaygın olarak kullanılan yöntemlerden biri olan Mc Cabe karmaşıklık ölçütü ve kod gözden geçirme teknikleri bu bölümde tanıtılmaktadır.



Gerçekleştirme Fazı

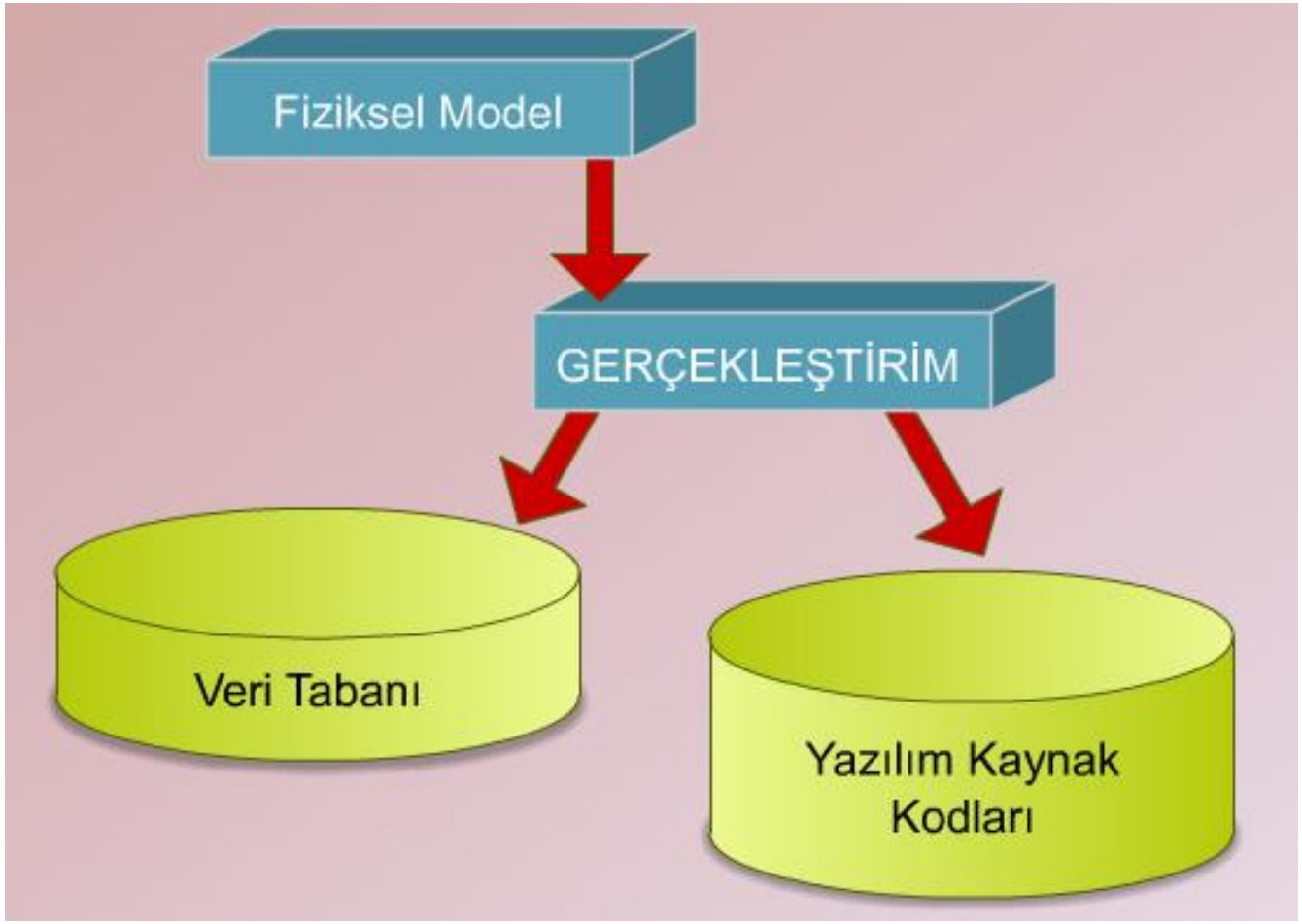
- Programlama Dilleri
- Yazılım Gerçekleştirme
- Derleyiciler
- Kodlama Etkinliği ve Kuralları
- Modüler Gerçekleştirme
- Gerçekleştirmenin Belgelendirilmesi

Giriş

- Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir.

- Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

Gerçekleştirim Çalışması



Yazılım Geliştirme Ortamları

- Programlama Dili
- Veri Tabanı Yönetim Sistemi
- Hazır Program Kitapçıkları
- CASE Araçları

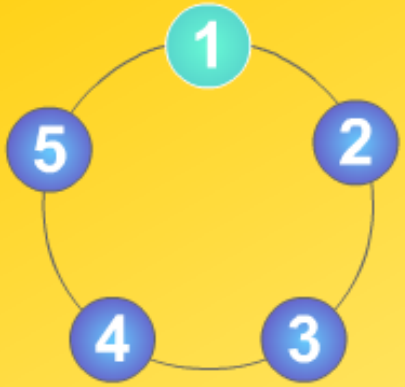
Programlama Dilleri

Geliştirilecek uygulamaya göre kullanılabilecek programlama dili de değişmektedir. Uygulama alanlarının bir sınıflaması:

- Veri işleme yoğunluklu uygulamalar
- Hesaplama yoğunluklu uygulamalar
- Süreç ağırlıklı uygulamalar
- Sistem programlamaya yönelik uygulamalar
- Yapay us (Y. zekâ- Y.Akıl) uygulamaları

biçiminde verilmektedir.

Uygulama Çeşitleri

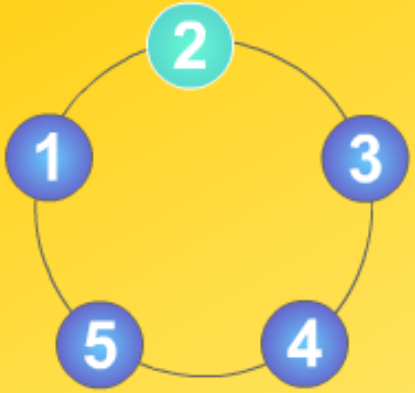


1

Veri İşleme Yoğunluklu Uygulamalar

Günümüzde, bilgi sistemi uygulamaları olarak bilinen ve yoğun veri işlemeyi gerektiren uygulamalar (Nüfus, Seçmen Kütükleri, Adli Sicil vb) için 90'lı yıllara kadar COBOL en yaygın olarak kullanılan programlama dili idi. 90'lı yılların başlarında gelişen ve veri tabanı bağlantılı çalışan Görsel Programlama Dilleri (Visual Basic, Delphi, Visual Age, Developer vb) bugünün uygulamalarında yaygın olarak kullanılmakta ve COBOL ve geçmişteki uygulamalar da bu dillere dönüştürülmektedir. Öte yandan Nesne-Kökenli Programlama dillerinin (C++, Java, Smalltalk vb) kullanımı giderek yaygınlaşmaktadır.

Uygulama Çeşitleri



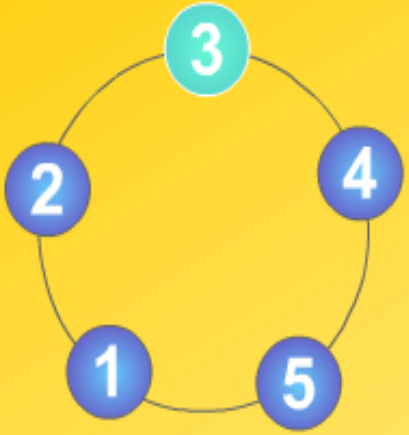
2

Yoğun Matematiksel ve Mantık İşlemi Gerektiren Uygulamalar

Bu uygulamalarda (NASA, uçak mühendisliği vb) program kütüphanelerinin zenginliği nedeniyle halen en yaygın olarak kullanılan programlama dilleri arasında, FORTRAN yer almaktadır. Yeni geliştirilen, bu tür uygulamalarda ağırlıklı olarak C programlama dili kullanılmaktadır.

Teknolojik gelişim, bu tür uygulamaları hızlandırmak amacıyla paralel işlem yapmayı olanaklı kılmış, bu bağlamda Paralel Programlama dilleri gelişmiştir. Ayrıca mevcut programlama dillerinin paralel sürümleri geliştirilmiştir (Paralel FORTRAN, Paralel C vb). Bu alanda kullanılan programlama dillerinin bir başka özelliği ise, yüksek duyarlıkta işlem yapmayı olanaklı kılmalarıdır.

Uygulama Çeşitleri

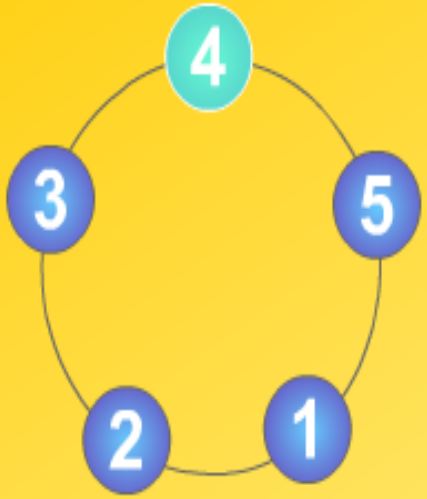


3

Süreç Ağırlıklı Uygulamalar

Bir başka deyişle "gerçek zamanlı" uygulamalarda, veri işleme ve matematiksel işlem yapma boyutlarının yanı sıra süreçler arası eşgüdüm, donanımın yazılım aracıyla sürülmesi boyutları önem kazanmaktadır. Bazı işletim sistemi yazılımları, uçak kontrol sistemleri, bankacılık sistemleri bu tür yazılımlara örnek olarak verilebilir. Bu tür uygulamalarda genellikle düşük düzey programlama dilleri (assembly dili) ve özel programlama dilleri ve henüz yaygınlaşmamış olmakla birlikte C programlama dili kullanılmaktadır.

Uygulama Çeşitleri

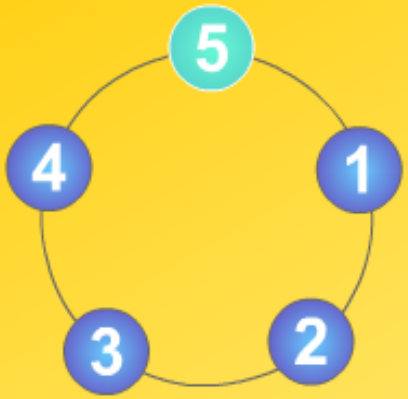


4

Sistem Programlamaya Yönelik Uygulamalar

Bilgisayarın yönetimine ilişkin uygulamalardır (İşletim sistemi vb). Bu tür yazılımlarda kullanılan programlama dili kapalı sistemler için assembly programlama dili, açık sistemler için ise, assembly dili ile birlikte C programlama dilidir.

Uygulama Çeşitleri



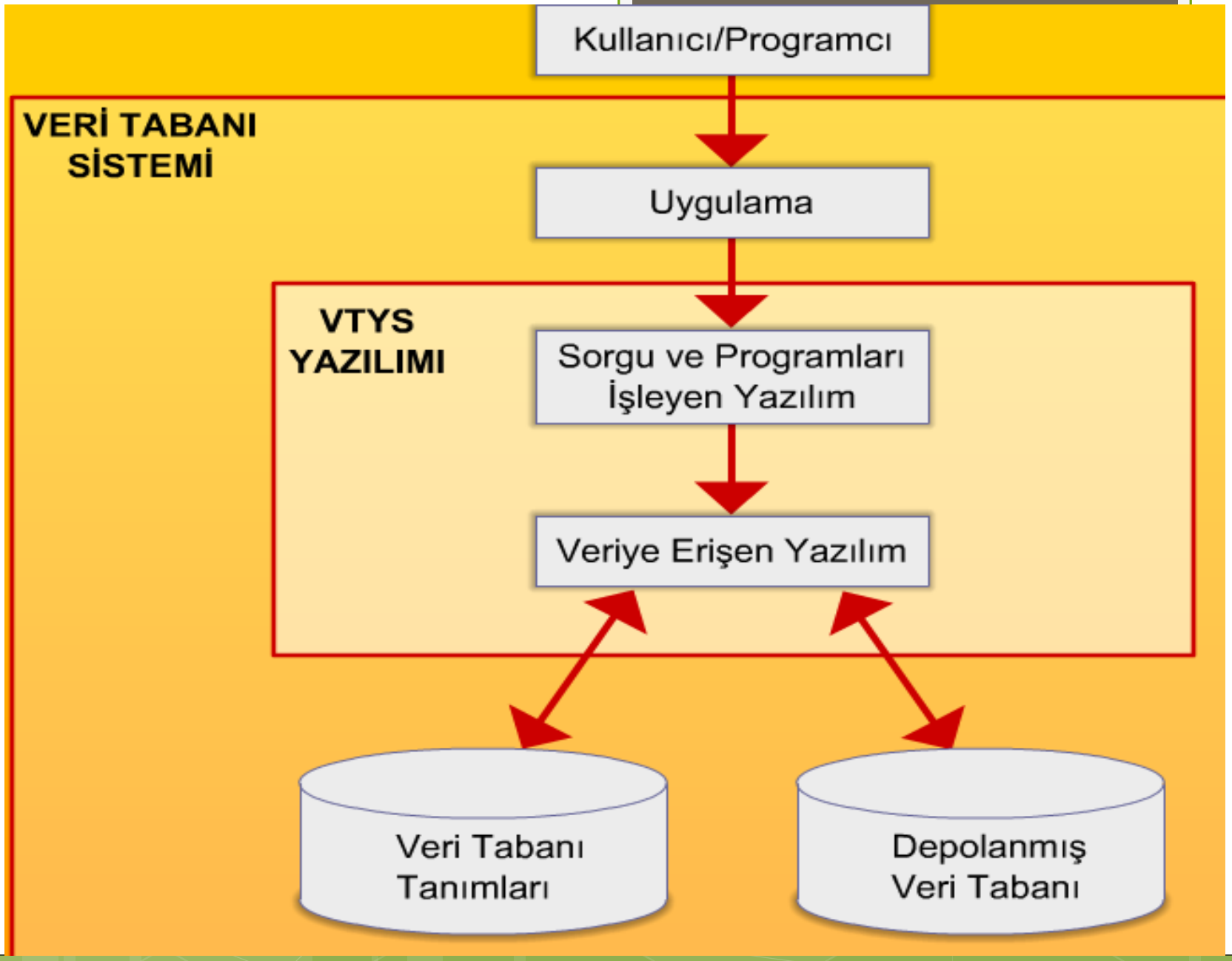
5

Yapay Us Uygulamaları

Kural-çıkarım ilkelerine dayalı uygulamalardır. Kuralların tanımlanması, bu kurallardan çıkarımlar yapılması ve çıkarımlardan yeni kurallar tanımlanması biçiminde etkileşimli olarak çalışmayı gerektiren uygulamalardır. Bu tür uygulamalarda en yaygın olarak kullanılan programlama dilleri ise Lisp ve Prolog'dur.

Veri Tabanı Yönetim Sistemleri

- Birbiri ile ilişkili veriler topluluğu veri tabanı olarak tanımlanmaktadır. Veri tabanı herhangi bir boyutta ya da karmaşıklıkta olabilir.
- Kişisel telefon rehberinizdeki adres bilgileri bir veri tabanı örneği oluşturduğu gibi, maliye bakanlığı bünyesinde saklanan ve vergi ödemesi gereken tüm kişilerin bilgilerinin saklandığı uygulama da bir başka veri tabanı örneğidir.
- Veri tabanını oluşturan veriler birbiriyle ilişkili verilerdir. Bir veritabanında veriler arası ilişkiler ile veri değerleri bulunur.


Kullanıcı/Programcı
VERİ TABANI SİSTEMİ
Uygulama
VTYS YAZILIMI
Sorgu ve Programları İşleyen Yazılım
Veriye Erişen Yazılım
Veri Tabanı Tanımları
Depolanmış Veri Tabanı

Veritabanı Yaklaşımının Geleneksel Kütük Kavramından Ayıran Özellikler

- 1. Veri Sözlüğü
- 2. Veri Soyutlama
- 3. Program-Veri ve Program-İşlem Bağımsızlığı
- 4. Birden Çok Kullanıcı Desteği
- 5. Verinin Birden Fazla İşlem Arasında Paylaşımı

Veri Sözlüğü

- Uygulama yazılımında kullanılan tüm veri tanımlarının yapısal ve ayrık bir biçimde saklanmasını sağlayan bir katalog bilgisi veya **veri sözlüğü**nün varlığı.

Dosya Adı	Değişken Adı	Veri Tipi	Uzunluk	Açıklama
Users	user_id	int	4	Kullanıcı Numarası (ID)
	nick	varchar	10	Kullanıcı Adı
	fullname	varchar	20	Kullanıcı Adı Soyadı
	email	varchar	20	Kullanıcı E-Mail Adresi
	password	varchar	8	Kullanıcı Şifresi
	accesslevel	int	1	Kullanıcı Yetkilendirme Seviyesi (0..9)
Codes	code_id	int	6	Kod Numarası (ID)
	language	varchar	15	Programlama Dil İsmi
	code_category	varchar	15	Kod Kategorisi
	add_date	datetime		Kod Ekleme Tarihi
	score	int	2	Kod Puanı (0-10)
	search_word	varchar	30	Kod Arama Kelimeleri
Forum	comments	longtext		Forum Yorumları
	forum_category	varchar	15	Forum içerisindeki dil kategorileri
	forum_date	datetime		Foruma bilgi giriş tarihi
Code_DB	VT_id	int	2	Veri Tabanı Numarası (ID)
	VT_code_lang	varchar	15	VT'deki Programlama Dilleri
User_DB	user_number	int	4	(Derived) Kullanıcı ID

Veri Soyutlama

- Veri tabanı yaklaşımının bir temel karakteristiği kullanıcıdan verinin saklanması konusundaki detayları gizleyerek veri soyutlamasını sağlamasıdır. Bu soyutlamayı sağlayan ana araç veri modelidir. Veri modeli şu şekilde tanımlanabilir.

Bir veri modeli bir veritabanının yapısını açıklamakta kullanılan kavramlar setidir.

Veri Soyutlama

- Bir veri tabanının yapısı ile veri tipleri, ilişkiler ve sınırlamalar kastedilmektedir. Pek çok veri modeli ayrıca veri tabanı üzerinde belirtilen getiri (çağrı) ve güncellemeler için temel işlemler setini içerir.
- Davranışı belirlemek üzere veri modelinin içine kavramları da eklemek mümkündür. Bu da, temel işlemlere ek olarak kullanıcı tarafından tanımlanmış işlemlerin de veri modeline eklenmesiyle mümkündür.
- Bir veri modelindeki genel işlemlere örnek olarak Ekle, Sil, Değiştir, Eriş verilebilir.

Program-Veri ve Program-İşlem Bağımsızlığı

- Geleneksel dosya işleme yönteminde, veri dosyalarının yapısı bu dosyalara erişim programları içine gömülmüştür. Bundan dolayı da dosyanın yapısındaki herhangi bir değişiklik bu dosyaya erişen tüm programlarda değişiklik yapılmasını gerektirir.
- Tersine VTYS erişim programları veri dosyalarından bağımsız olarak tasarlanmışlardır. VTYS de saklanan Veri dosyalarının yapısı erişim programlarından ayrı olarak katalogda tutulduğundan program ve veri bağımsızlığı sağlanır.

Program-Veri ve Program-İşlem Bağımsızlığı

ÖRNEK

Örneğin bir veri dosyasına yeni bir alan eklenmek istendiğinde, bu veri dosyasını kullanan tüm programlara bu alanın eklenmesi gerekir. VTYS yaklaşımında ise sadece katalogdaki veritabanı tanımlarına bu alan eklenerek bu sorun çözülebilir.

Birden Çok Kullanıcı Desteği

- Bir veri tabanının birçok kullanıcısı vardır ve bunların her biri veri tabanının farklı bir görüntüsüne gereksinim duyabilir. Bir görüntü veri tabanının alt kümesi olabilir veya veri tabanından elde edilmiş fakat kesin olarak saklanmamış sanal veri içerebilir.
- Bazı kullanıcılar, kullandıkları verilerinin veri tabanından türetilmiş veriler mi? Yoksa veri tabanında gerçekte saklanan veriler mi olup olmadığı ile ilgilenmezler. Kullanıcıları farklı uygulamalara sahip çok kullanıcılı bir VTYS çoklu görüntü tanımlama olanaklarını sağlamalıdır.

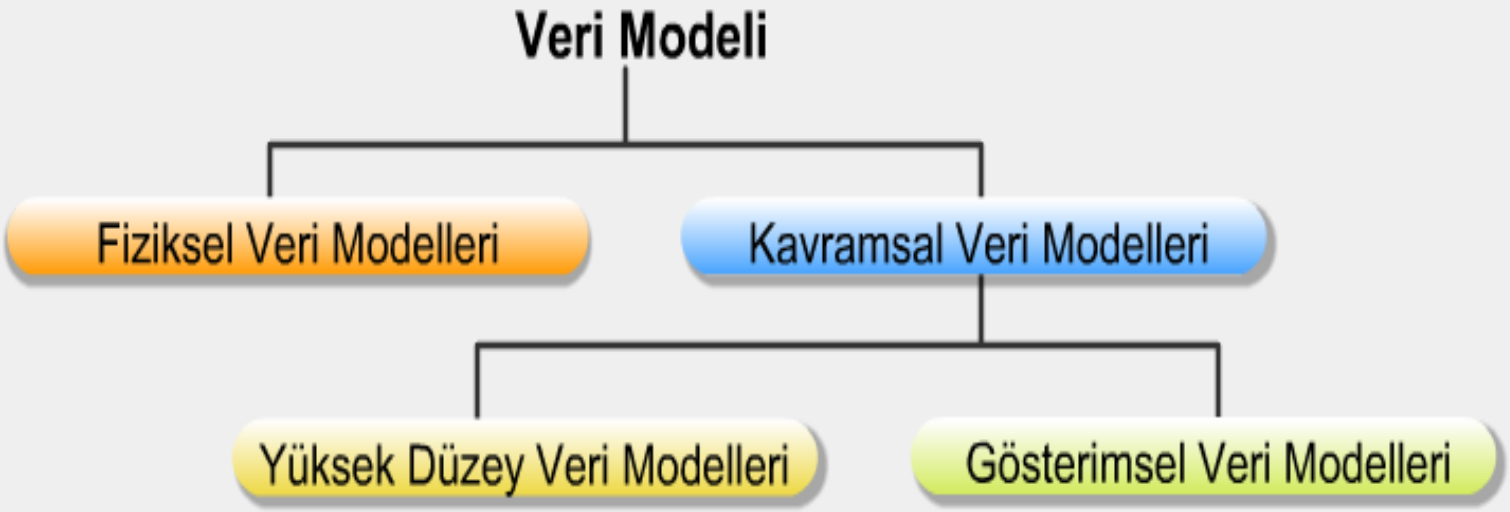
Verinin Birden Fazla İşlem Arasında Paylaşımı

- Çok kullanıcılı VTYS yazılımlarının temel rolü eş zamanlı işlemlerin karışıklık olmadan doğru bir şekilde yapılmasına olanak tanımak işlemlerin doğru olarak yapılmasını garanti etmektir.
- Bu özellik veri tabanı (VT) kavramını, dosya işleme kavramından ayıran en önemli özelliktir. Bu kontrol aynı anda birden çok kullanıcının aynı veriyi güncellemeye çalışmasını, güncellemenin doğru olması açısından garanti eder.

VTYS Kullanımının Ek Yararları

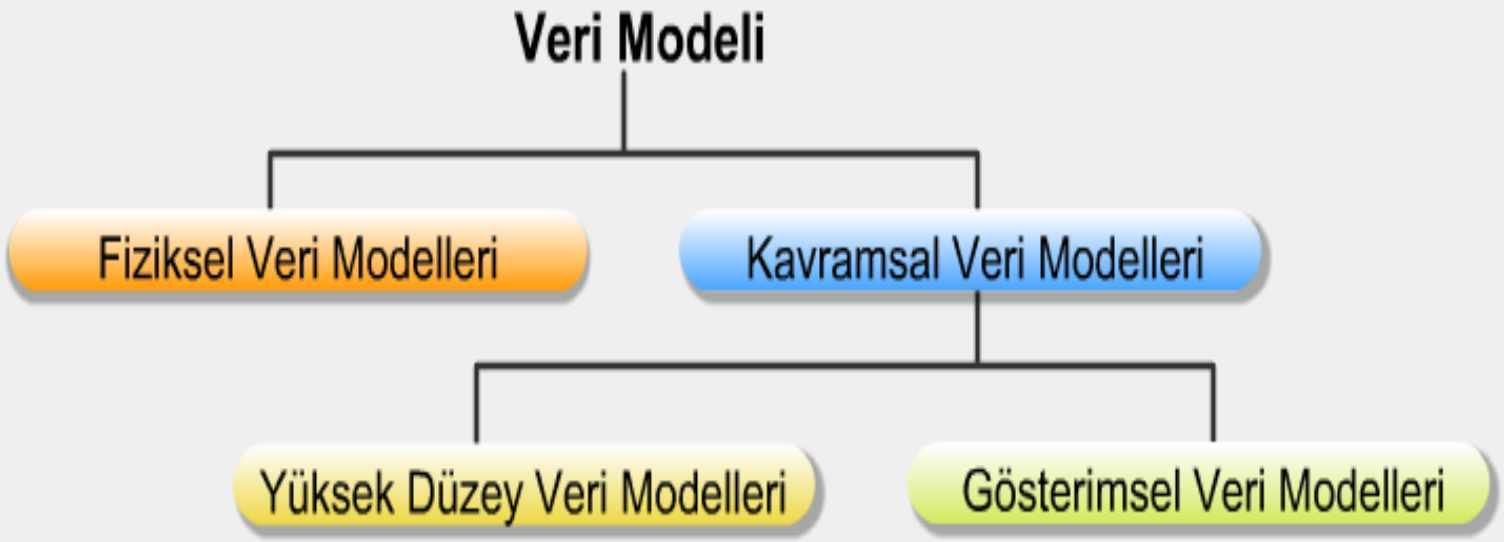
1. Genişleme potansiyeli,
2. Esneklik,
3. Uygulama geliştirme zamanının azalması,
4. Güncel bilgilerin tüm kullanıcılara aynı zamanda ulaşması,
5. Ölçümde ekonomi,
6. İşletme ortamındaki ortak verilerin tekrarının önlenmesi verilerin merkezi denetiminin ve tutarlılığının sağlanması,
7. Fiziksel yapı ve erişim yöntemi karmaşıklıklarının Her kullanıcıya yanlız ilgilendiği verilerin kolay anlaşılır yapılarda sunulması,
8. Uygulama yazılımı geliştirmenin kolaylaşması,
9. Fiziksel yapı ve erişim yöntemi karmaşıklıklarının Her kullanıcıya yanlız ilgilendiği verilerin kolay anlaşılır yapılarda sunulması,

Veri Modelleri



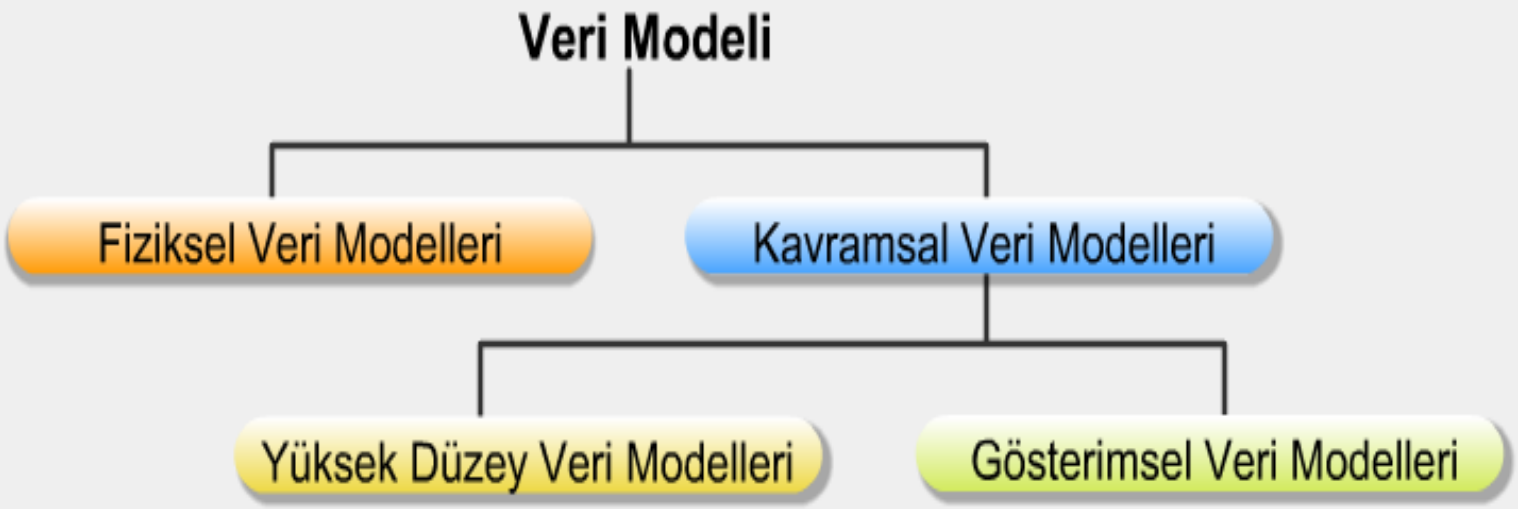
Fiziksel veri modelleri ise verinin bilgisayarda nasıl saklandığının detayları ile ilgili kavramları sağlarlar. Bu ikisinin arasında ise gösterimsel veri modelleri vardır. Bu modeller bilgiyi kayıt biçemi, sırası ve erişim yollarıyla göstererek verinin bilgisayar ortamında nasıl saklandığını açıklar. Burada erişim yolu, belirli bir veritabanı kaydını etkin bir şekilde aramayı sağlayan yapıdır.

Veri Modelleri



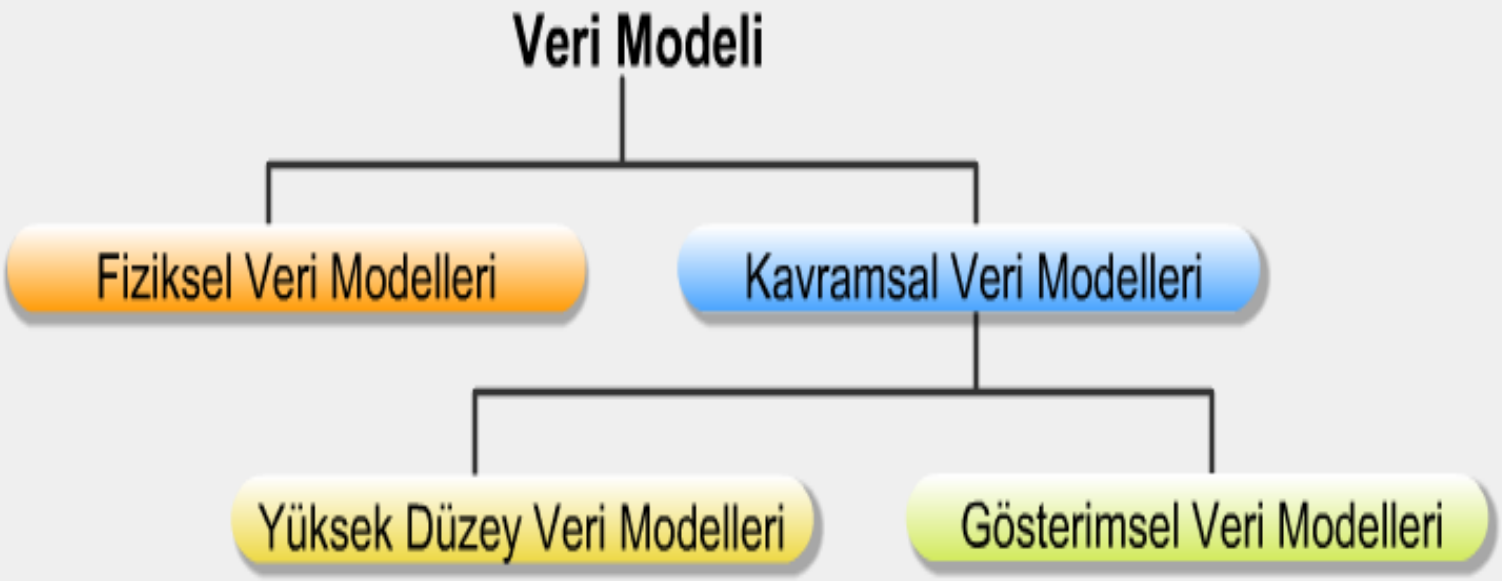
Bunlar hem kullanıcı tarafından anlaşılan hem de verinin bilgisayar içerisindeki gösteriminden çok da fazla uzak olmayan kavramları sağlarlar.

Veri Modelleri



Yüksek düzey veri modelleri varlık, özelik ve ilişki gibi kavramları kullanır. Varlık veri tabanında saklanan ve gerçek dünyadan bir nesne veya kavramdır; proje, işçi gibi. Özelik, varlığı anlatan bir özelliği gösterir işçinin adı, ücreti gibi. Bir veya daha fazla varlık arasındaki ilişki ise bir işçi ve proje arasındaki çalışma ilişkisidir.

Veri Modelleri



Gösterimsel veri modelleri, ticari veri tabanlarında sıklıkla kullanılır ve belli başlı veri üç modeli içerir. İlişkisel, ağ ve hiyerarşik veri modeli. Nesne yönelimli veri modelleri daha yüksek seviyeli gerçekleştirim veri modelleridir ve kavram veri modeline daha yakındır.

Şemalar

- Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önemlidir. Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlandırılır. Veri tabanı şeması, tasarım sırasında belirtilir ve sıkça değişmesi beklenmez.

- Pek çok veri modeli şemaları, diyagramlar halinde göstermek için belli gösterim biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek örneğini göstermez.

VTYS Mimarisi

İçsel Düzey

Veritabanının fiziksel saklanma yapısını açıklar. Fiziksel veri modeli kullanır ve veritabanına erişim yolu ile veri saklamanın tüm detaylarını açıklar.

Kavramsal Düzey

Kavramsal düzey ise kavramsal şema içerir ve kullanıcılar için veritabanının yapısını açıklar. Gerçek fiziksel yapının detaylarını kullanıcıdan gizler, veri tipleri, varlıklar ilişkiler, kullanıcı işlemleri ve sınırlamalar üzerine konsantre olmamızı sağlar. Daha yüksek seviyede veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Dışsal Düzey

Bu düzey, dış şemalar veya kullanıcı görüşleri içerir. Her dış şema veritabanının bir bölümünü açıklar ve her gruba kendi ilgilendiği görüşü sunarken, diğer bir gruptan ilgilenmediği görüşü saklar. Daha yüksek düzeyde veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Veri Tabanı Dilleri ve Arabirimleri

Veri Tanımlama Dili (VTD)	Kavramsal şemaları tanımlamak üzere veritabanı yönetici ve tasarımcısı tarafından kullanılır
Saklama Tanımlama Dili (STD)	İçsel şemayı tanımlamak için kullanılır.
Görüş Tanımlama Dili (GTD)	Görüş tanımlama dili kullanıcı görüşlerini tanımlamak ve kavramsal şemaya dönüştürmek amacıyla kullanılır.
Veri İşleme Dili (VİD)	Veri işleme dilidir. Veri tabanı oluşturulduktan sonra Veri tabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır.

Veri Tabanı Sistem Ortamı

- VTYS, karmaşık bir yazılım sistemidir. Burada bir VTYS'yi oluşturan yazılım bileşenlerden söz edilmektedir.
- Tipik bir VTYS yazılımı bileşenleri, VTYS Kataloğu, veri tabanı derleyicisi (VTD), veri yöneticisi, VİD derleyicisi, VT işlemcisi ve yardımcı yazılımlar biçimindedir.

VTYS'nin Sınıflandırılması

- Bir VTYS'ni sınıflandırmada kullanılan temel kriter VTYS'nin dayandığı veri modelidir.

- En fazla kullanılan veri modelleri ilişkisel, ağ, hiyerarşik, nesne-yönelimli ve kavramsal modellerdir.

Hazır Program Kitaplıkları

- Hemen hemen tüm programlama platformlarının kendilerine özgü hazır program kitaplıkları bulunmaktadır.
- Söz konusu kitaplıklar, hemen hemen tüm uygulamalarda kullanılabilecek ortak program parçaları (OCX,VBX vb), sistem yönetimine ilişkin yazılımları içerir.
- Söz konusu yazılımlar program geliştirme hızını oldukça arttırmaktadır. Günümüzde bu tür kitaplıkların edinilmesi, internet sayesinde oldukça kolaylaşmıştır.

CASE Araç ve Ortamları

- Günümüzde bilgisayar destekli yazılım geliştirme ortamları (CASE) oldukça gelişmiş durumdadır.
- CASE araçları, yazılım üretiminin hemen her aşamasında (planlama-çözümleme-tasarım-gerçekleştirim-sınama) üretilen bilgi ya da belgelerin bilgisayar ortamında saklanmasını, bu yolla kolay erişilebilir ve yönetilebilir olmasını olanaklı kılar.
- Hemen tüm CASE araçları, belirli bir standarda uygun olarak geliştirme yapmayı zorlar. Bu yolla yapılan üretimin yüksek kalitede olması sağlanır. CASE araç ve ortamları Bölüm 12'de ayrıntılı olarak incelenmiştir.

Kodlama Stili

- Yazılım Kod stili konusunda herhangi bir kabul edilmiş standart bulunmamaktadır.
- Bu konuda, yazılım geliştiren ekiplerin, kodlama aşamasına başlamadan kodların düzeni konusundaki standartlarını ya da kurallarını geliştirmeleri ve bu kuralların uygulamaya geçmesini sağlamaları önerilmektedir.

Açıklama Satırları

- Bir program parçasını anlaşılabilir kılan en önemli unsurlardan biri bu program kesiminde içerilen açıklama satırlarıdır. Her bir program modülü içerisine

- Modül başlangıç açıklamaları
- Modül Kod Açıklamaları
- Boşluk satırları

eklenmelidir.

Açıklama Satırları

```
/*****************************************************************
*  ©
*****************************************************************
*  Modül Adı:
*  Kütük Adı:
*  İşlevi:
*  Desteklenen Platformlar:
*  Desteklenen Derleyiciler:
*  Taşınabilirlik Konuları:
*  Kullanılan İşlevler:
*  Hatalar:
*  Bilinen Düzeltilmemiş Yanlışlar:
*  Programcı:
*  Tarihçe Günlüğü:
*  İsim                    Tarih                    Açıklama
*
*****************************************************************
```

Kod Biçimlemesi

- Ancak Program kodlama düzeni ise gerçekleştirim aşamasında çözümlenir.
- Programların okunabilirliğini arttırmak, anlaşılabilirliğini kolaylaştırmak amacıyla açıklama satırları kullanımının yanı sıra belirli bir kod yazım düzeni kullanılması gerekmektedir

```
DO I=1 TO N-1;T=I; DO J=I TO I+1;IF A(J)< A(T) THEN DO
T=J;
END;
IF T<>I THEN DO H=A(T);A(T)=A(I); A(I)=T;END;END;
```

(a) Kötü stilde kodlanmış program parçası

```
DO I = 1 TO N-1;
   T=I;
   DO J = 1 TO I+1;
   IF A(J) < A(T) THEN DO
      T=J;
   END;
   IF  T <> I  THEN DO
      H = A(T);
      A(T) = A(I);
      A(I) = H;
   END;
END;
```

(b) İyi stilde kodlanmış program parçası

Anlamlı İsimlendirme

- Hangi değişkenlerin hangi veri tabanı tablosuna ilişkin oldukları,
- Hangi değişkenlerin klavye ve ekran aracılığı ile değer aldıkları
- Hangi değişkenlerin yazıcı çıktılarında görülmek üzere kullanıldıkları,
- Hangi değişkenlerin yalnızca ilgili kod içerisinde ara değişkenler olarak kullanıldıkları

Tekrar

Yapısal Programlama Yapıları

- Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve program karmaşıklığının azaltılması amacıyla "yapısal programlama yapıları" kullanılarak yazılması önemlidir.
- Yapısal Programlama Yapıları, temelde, içinde "go to" deyimi bulunmayan, "tek giriş ve tek çıkışlı" öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır.

Yapısal Programlama Yapıları

Ardışıl İşlem Yapıları

Ardışıl işlemler, herhangi bir koşula bağlı olmaksızın birbiri ardına uyglanması gereken işlemler olarak tanımlanır. Hemen her tür programlama dilinde bulunan, aritmetik işlem deyimleri, okuma/ yazma deyimleri bu tür yapılara örnek olarak verilebilir.

Yapısal Programlama Yapıları

Koşullu İşlem Yapıları

Üç tür Koşullu işlem yapısı bulunmaktadır: tek koşullu işlem yapısı (if-then), iki koşullu işlem yapısı (if-then-else) ve çok koşullu işlem yapısı (case-when). 70'li yılların ortalarından sonra gelen programlama dillerinin hemen hepsinde, bu yapılar doğrudan desteklenmektedir.

Yapısal Programlama Yapıları

Döngü Yapıları

Döngü yapıları, belirli bir koşula bağlı olarak ya da belirli sayıda, bir ya da daha çok kez yinelenecek işlemler için kullanılan yapılardır. Temelde üç tür döngü yapısı bulunmaktadır:

- Belirli sayıda yinelenecek işlemler için kullanılan yapılar (for yapısı),
- Bir koşula bağlı olarak, sıfır ya da birden çok kez yinelenecek işlemler için kullanılan yapılar (while-end yapısı),
- Bir koşula bağlı olarak, bir ya da daha çok kez yinelenecek işlemler için kullanılan yapılar (repeat-until yapısı).

Bu yapıların her biri "tek girişli ve tek çıkışlı" yapılardır. Programlar, yalnızca bu yapılar kullanılarak kodlandığında ve uygun açıklama satırları ile desteklendiğinde, program bakımı kolaylaşmakta ve geliştirme hızlanmaktadır.

PROGRAM KARMAŞIKLIĞI

- Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür.
- Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.

1. Programın Çizge Biçimine Dönüştürülmesi

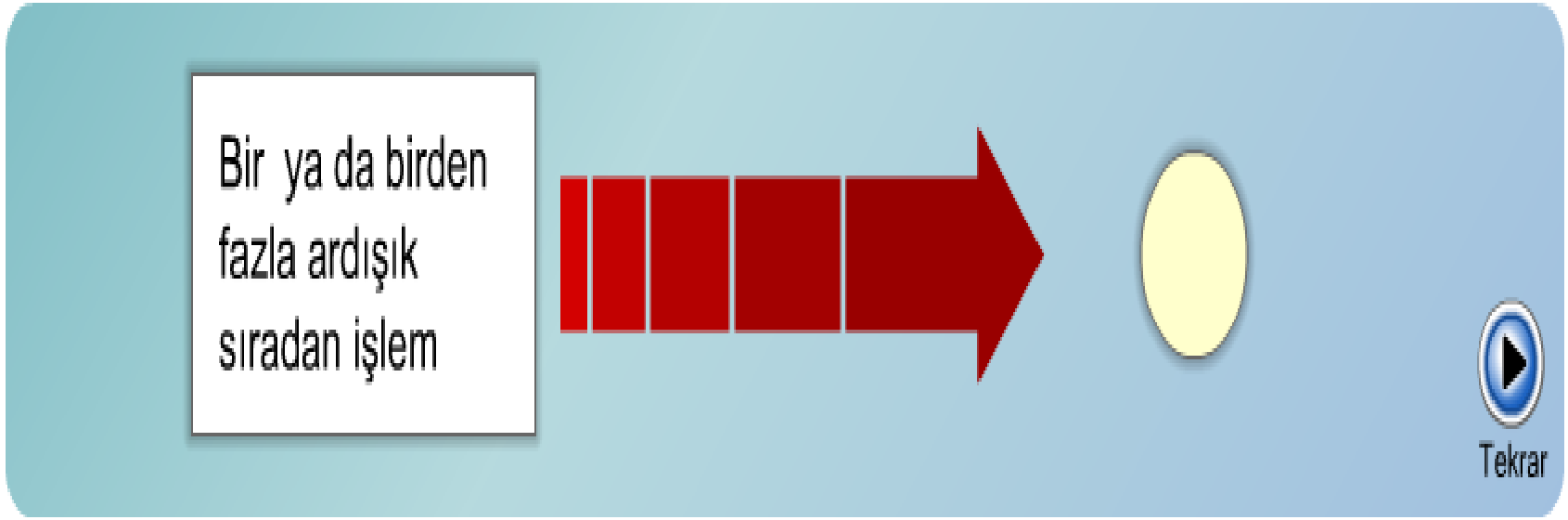
2. Mc Cabe Karmaşıklık Ölçütünün Hesaplanması

Programın Çizge Biçimine Dönüştürülmesi

- Bir program bir ana program ve onunla ilgili birden fazla alt programdan oluşur.
- Gerek ana program gerekse alt programların tümü, McCabe Karmaşıklık ölçütünün hesaplanmasından önce çizge biçimine dönüştürülmelidir.
- Bir program parçasının çizge biçimine dönüştürülmesi için gerekli kurallar aşağıda açıklanmaktadır.

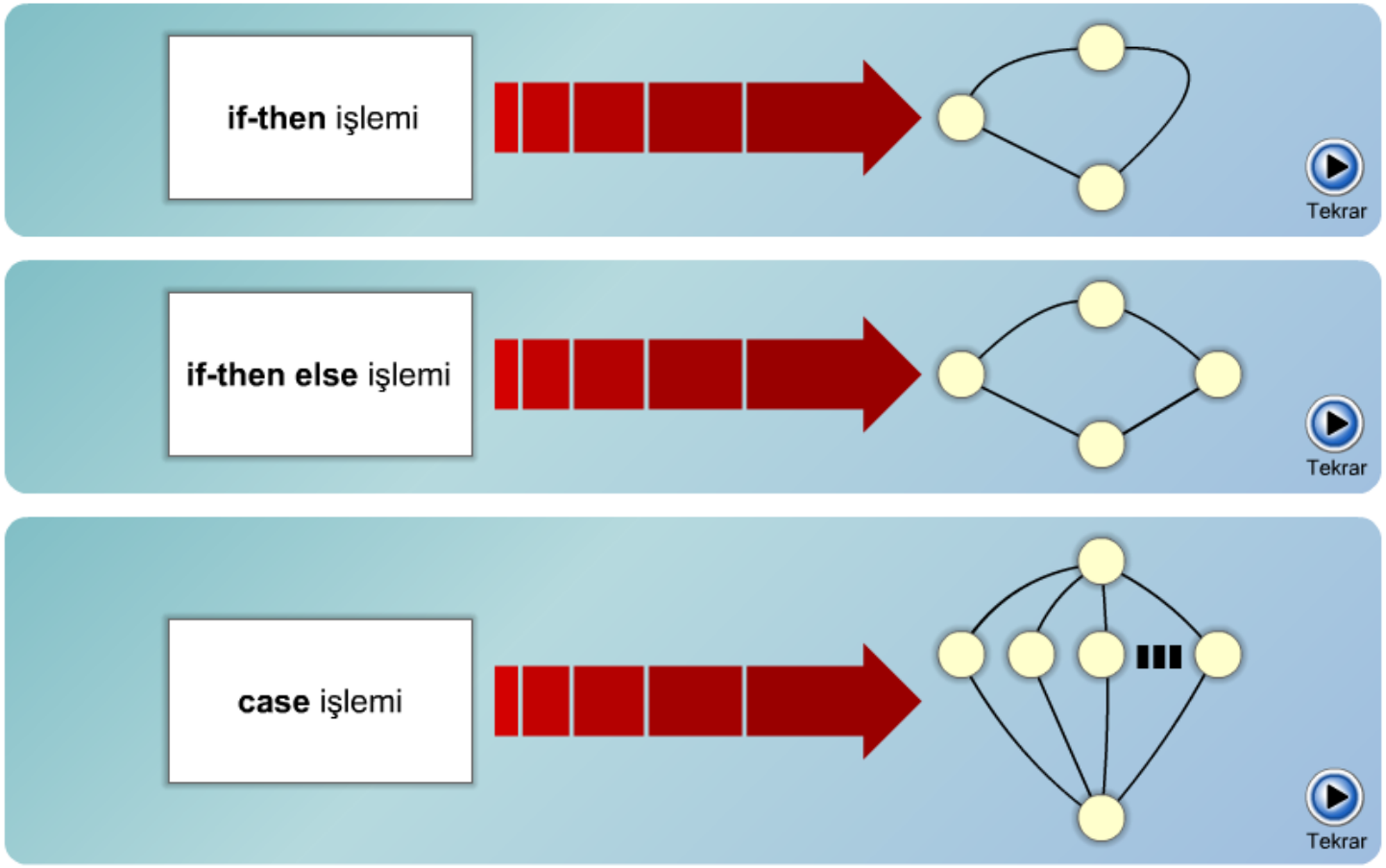
Programın Çizge Biçimine Dönüştürülmesi

1. Sıradan İşlemler: Bir sıradan işlem ya da birden fazla ardışık sıradan işlem çizgede bir düğüme dönüştürülür.



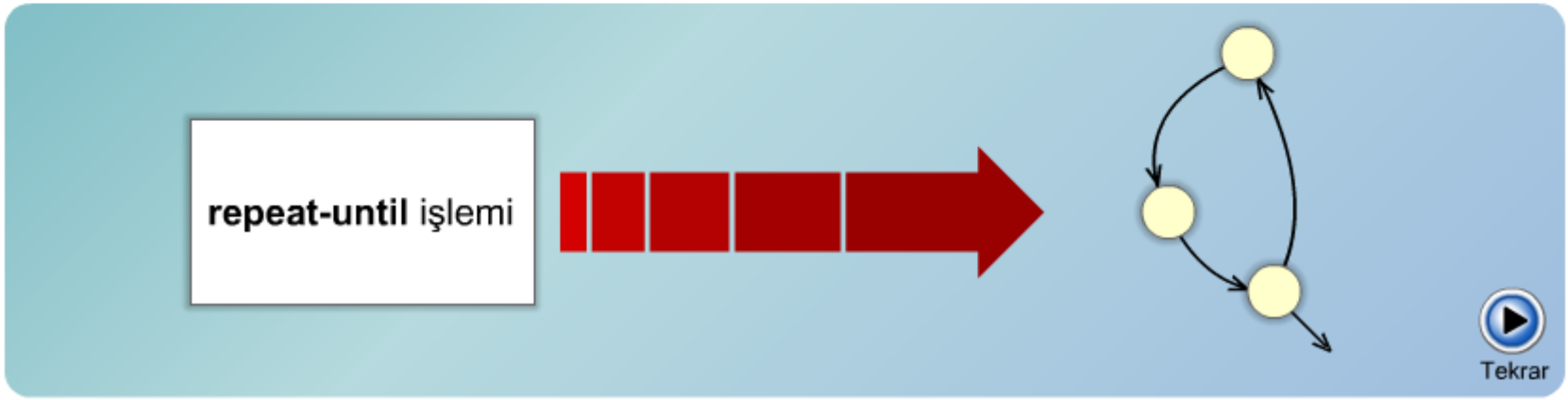
Programın Çizge Biçimine Dönüştürülmesi

2. Koşullu İşlemler:



Programın Çizge Biçimine Dönüştürülmesi

3. Döngü İşlemleri:



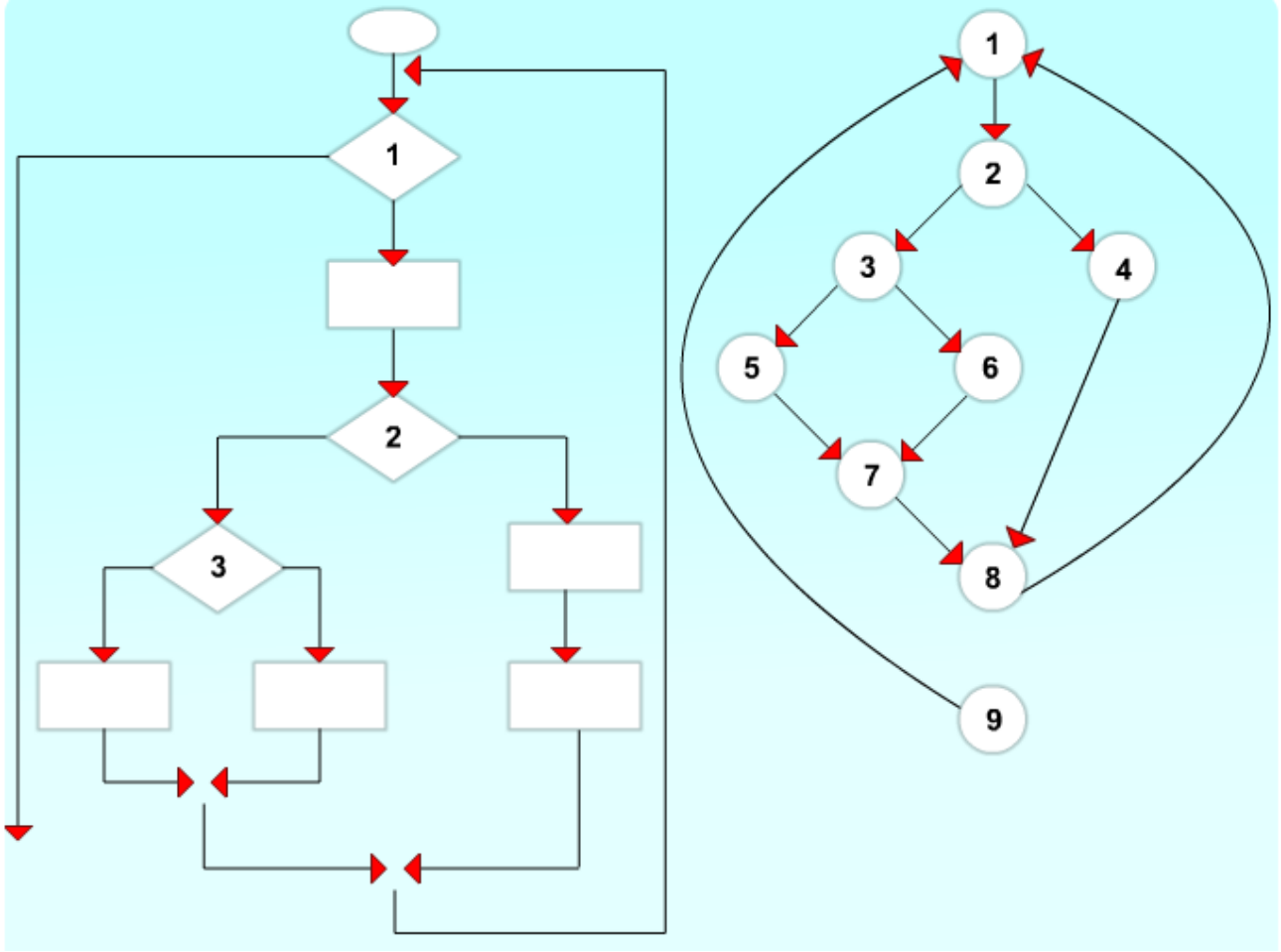
McCabe Karmaşıklık Ölçütü Hesaplama

- Ana program ve alt programlar çizge biçimine dönüştürüldükten sonra programın McCabe karmaşıklık ölçütü (V(G)) aşağıdaki biçimde hesaplanır.

$$V(G) = k - d + 2p$$

- k: kenar sayısı
- d: düğüm sayısı
- p: bileşen sayısı

McCabe Karmaşıklık Ölçütü Hesaplama



McCabe Ölçütü

- Şekil 6.5 (a) da bir programın iş akış şeması verilmekte, (b) kısmında ise önceki kesimde verilen kurallar uygulanarak programın çizge biçimine dönüştürülmüş biçimi verilmektedir. McCabe Karmaşıklığını hesaplamak için kenar ya da dallar ve düğümler sayılıp formülün uygulanması gerekir. Bu örnekte:

- **k = 11** Kenar sayısı
  d = 9 Düğüm sayısı
  p = 1 Bileşen sayısı

karmaşıklık: $V(G) = 11 - 9 + 2 \times 1 = 4$

OLAĞANDIŞI DURUM ÇÖZÜMLEME

- Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.
- Genelde kabul edilen kural, bir programın işletiminin sonlandırılması işleminin -elektrik kesintisi vb donanım hataları dışında- bütünüyle program denetiminde olmasıdır. Bu kural olağan dışı sonlandırma işlemini de kapsamaktadır.
- Bu nedenle program kodları oluşturulurken, olağandışı durumların da dikkate alınması ve bu durumlardaki program davranışına ilişkin yöntemler geliştirilmesi gerekmektedir.

Ada Dili Olağandışı Durum Çözümleme Yapısı

```
begin
     . . . .
     . . . .
     . . . .          Program kod satırları
     . . . .
     . . . .
     . . . .
exception
     when CONSTRAINT_ERROR  =>      SINIR_HATASI;
. .
     when TABLO_DOLU        =>      . . .
end;
```

Olağandışı Durum Tanımları

- Olağan dışı durumlar, programlama dili tarafından tanımlı durumlar olduğu gibi kullanıcı tarafından da tanımlanabilmektedir.

- Yukarıdaki örnekte, CONSTRAINT_ERROR adlı olağan dışı durum ADA Programlama dili tarafından tanımlanmış, TABLO_DOLU adlı olağan dışı durum ise programcı tarafından tanımlanmıştır.

Olağandışı Durum Tanımları

Bu tür olağan dışı durumlar, olağan dışı durumu tanımlayan ve sonucunda "doğru" ya da "yanlış" değeri üreten biri mantıksal yordam ya da fonksiyon tanımından oluşmaktadır.

Şekilde ADA Programlama dili platformunda kendiliğinden tanımlı olağandışı durum örnekleri verilmektedir.

Farklı Olağandışı Durum Çözümleme Yaklaşımları

Anında Durdurma:

Hata bulunduğunda program açıklayıcı bir ileti vererek sona erer. Bu iletinin yanı sıra onaltılı sistemde bir döküm de verir. MS Office paketlerinde alınan "This Program Performed an illegal operation" iletisi ve ardından programın durması, bu tür olağandışı durum çözümleme yaklaşımına örnek olarak verilebilir. Bu yaklaşımda, verilen hata iletisinin anlamlı olması ve programcıya düzeltebilmek amacıyla yol gösterici olması önemlidir.

Hata Kodu Verme:

İlgili program modülünden başarısız şekilde çıkıldığında, programın bir hata kodu vermesi biçminde bir yaklaşımdır. Söz konusu kod "başarılı" ya da "başarısız" durumu belirten mantıksal bir kod olabileceği gibi, bazı uygulamalarda sayısal ya da metin biçiminde olabilir. Bu yaklaşımda, ilgili programında hata oluştuğu anlaşılabilmekte ancak hatanın hangi deyimde oluştuğu anlaşılamamaktadır.

Farklı Olağandışı Durum Çözümleme Yaklaşımları

Tanımlı Olağandışı Yordam Çalıştırma:

Programlama dili platformunun tanımladığı olağan dışı durum çözümleme olanaklarını kullanmak biçimindedir. COBOL Programlama dilindeki "ON READ ERROR" deyimi gibi.

Hata Yordamı Çalıştırma:

Olmayacak Değer Döndürme: Yordamın beklenmeyen bir değer geri döndürmesi gibi bir yaklaşımdır. Örneğin yaş hesaplaması yapıp, yaş bilgisini geri döndüren bir programın eksi değer döndürmesi gibi.

Tekrar

KOD GÖZDEN GEÇİRME

- Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeden okunabilir bir program yazamaz.
- Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işletime alınmamalıdır.
- Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir.

Gözden Geçirme Sürecinin Düzenlenmesi

- Gözden geçirme sürecinin temel özellikleri;
- Hataların bulunması, ancak düzeltilmemesi hedeflenir,
- Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullanılmasıdır.

Gözden Geçirme Sürecinin Düzenlenmesi

- Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.
- Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır.
- Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir. Gözden Geçirme çalışmasının olası çıktıları:

Gözden Geçirme Sürecinin Düzenlenmesi

- Programı olduğu gibi kabul etmek
- Programı bazı değişiklerle kabul etmek
- Programı, önerilen değişikliklerin yapılmasından sonra tekrar gözden geçirmek üzere geri çevirmek.

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.1. Öbek Arayüzü

1. Her öbek tek bir işlevsel amacı yerine getiriyor mu?

2. Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?

3. Öbek tek giriş ve tek çıkışlı mı?

4. Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.2. Giriş Açıklamaları

1. Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?

2. Giriş açıklama satırları, öbeğin amacını açıklıyor mu?

3. Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?

4. Giriş açıklama satırları, çıktıları (parametre, kütük vb) ve hata iletilerini tanımlıyor mu?

5. Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?

6. Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?

7. Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?

8. Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?

9. Her paragrafı açıklayan kısa açıklamalar var mı?

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.3. Veri Kullanımı

1. Işlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?

2. Değişken adları,işlevlerini yansıtacak biçimde anlamlı mı?

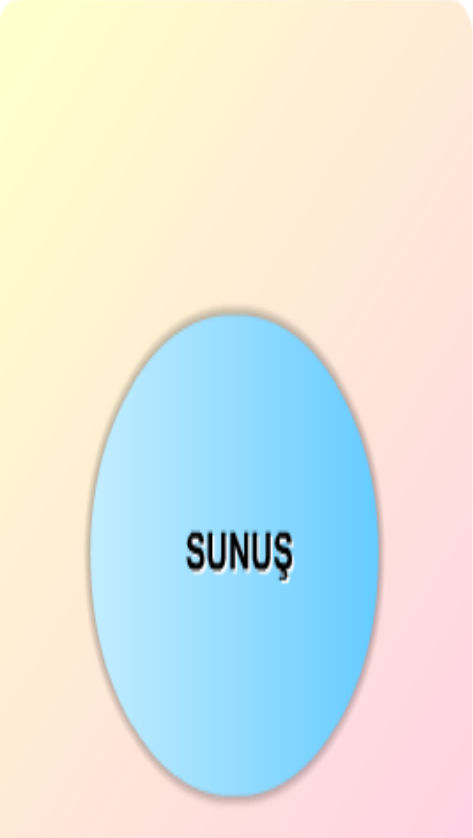
3. Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?

4. Her değişken tek bir amaçla mı kullanılıyor?

5. Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?

6. Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

Gözden Geçirme Sırasında Kullanılacak Sorular



6.6.2.5. Sunuş

1. Her satır, en fazla bir deyim içeriyor mu?

2. Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?

3. Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?

4. Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?

5. Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?

6. Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

Gerçekleştirme Öznitelikleri

- Etkinlik
- Aykırı Durumların Kotarılması
- Hata Ayıklama
- Atık/Çöp toplama
- Soyutlama
- Bilgi Gizleme
- Etiketleme
- Güvenlik
- Basitlik
- Belirgin Arayüz
- Taşınabilirlik
- Çok düzeyli koruma
- Gözden geçirme
- Belgelendirme

Alınan Dersler 1

- Bakım Programcısı kavramı uygulamada çalışmamakta, genellikle yazılımı geliştiren, bakımını da yapmakta ve usta-çırak ilişkisi içinde işler devredilmeye çalışılmaktadır.
- Bu durum, zaten kısıtlı olan bilişim insan gücü kaynağının verimsiz olarak kullanılmasına neden olmakta ve "kişiye bağlı" yazılımlar geliştirilmektedir.

Alınan Dersler 2

- Özellikle C programlama dili için programlama stili çok önemlidir. Sözdizim tanımları gereği C programları, okunmaları ve anlaşılmaları kolay olmayan programlardır.
- Bu nedenle, kodlamaya başlanılmadan önce, bu konuya ilişkin kullanacağınız standartları oluşturulmalı ve kodlama sırasında bu standartlara uyulup uyulmadığını denetlemeniz gerekmektedir.

Alınan Dersler 3

- Açıklama satırları ile ilgili kod uyumlu olmalı. Kod satırları arasında, süreç içerisinde yapılan değişiklikler, yeni eklentiler açıklama satırlarına yansıtılmamakta, bu durum, programların okunabilirliğini giderek zorlaştırmaktadır.

Alınan Dersler 4

- Süreç içinde anlamlı isimlendirmeler kaybolabilmektedir. Program bakımı, belli bir sistematik içerisinde yapılmamakta ve izlenmemektedir.
- Programların, süreç içerisinde, "yamalı bohça" ya dönüşmesi söz konusudur. Bir sistematik ve izleme olmadığında, bakım giderek zorlaşmaktadır.

Alınan Dersler 5

- Gözden Geçirme çalışmasının yapılması amacıyla yönetimi ikna etmek zor.
- Bu tür bir örgütlenmeye sıcak bakılmamakta, öte yandan, programcılar da benzer tepkiyi vermekte ve "sanki kendilerini denetliyorlarmış" mantığı ile davranmaktalar.
- Oysa, Gözden geçirme çalışması yapılmadığında, özellikle büyük yazılım projelerinin geliştirilmesi olanaksız.

Sorular

1. Kendi yazılım geliştirme ortamınızı açıklayınız.

2. Veri tabanı ile veri tabanı yönetim sistemi arasındaki farkı belirtiniz.

3. Veri tabanı Yönetim Sistemi kullanarak, uygulama geliştirme zamanının hasıl kısalacağını açıklayınız.

4. Veri tabanı Yönetim Sistemi kullanımının yararlarını ve aksak yönlerini belirtiniz?

5. Kullandığınız bir veri tabanı yönetim sistemini inceleyiniz. VTD, STD, GİD, GTD dillerinin özelliklerini araştırınız.

6. Kendi kullanımınız için kodlama stili geliştiriniz.

7. Kodlama stilleri ile programlama dilleri arasındaki ilişkiyi belirtiniz.

8. Bildiğiniz bir programlama dilinin, yapısal programlama yapılarından hangilerini doğrudan desteklediğini, hangilerini desteklemediğini araştırınız. Desteklenmeyen yapıların, bu programlama dilinde nasıl gerçekleştirilebileceğini belirtiniz.

9. Verilen bir N doğal sayısının asal olup olmadığını belirleyen bir programı dört değişik biçimde yazınız. Programlar arasındaki zaman farklılıklarını ölçünüz.

10. Program geliştirirken kullandığınız olağan dışı durum çözümleme yöntemlerini açıklayınız?

11. Geliştirdiğiniz bir program için, bölüm içerisinde verilen gözden geçirme sorularını yanıtlayınız. Programınızın niteliği hakkında ne söyleyebilirsiniz?

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

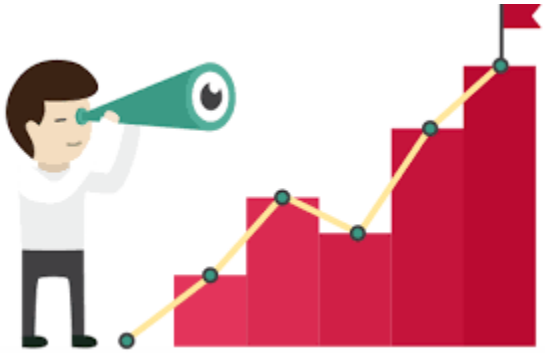
  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ 8.Hafta

Yazılım Doğrulama ve Geçerleme

Bölüm Hedefi

- Yazılım üretimi boyunca, "Doğru Yazılımı mı üretiyoruz?" ve "Yazılımı doğru olarak üretiyor muyuz?" sorularının yanıtlarını araştıran doğrulama ve geçerleme yöntemleri bu bölümde açıklanmaktadır.



Giriş

- Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.

- Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.

Giriş

- Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.
- Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklenmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.

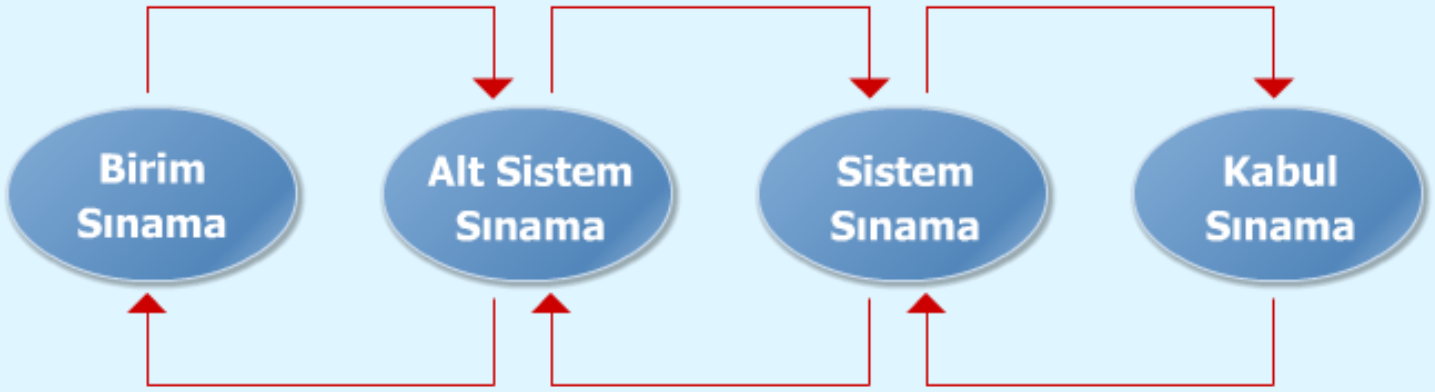
Doğrulama / Geçerleme

Doğrulama	Doğru ürünü mü üretiyoruz?	Ürünü kullanacak kişilerin isteklerinin karşılanıp karşılanmadığına dair etkinliklerden oluşur.
Gerçekleme	Ürünü doğru olarak mı üretiyoruz?	Ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinden oluşur.

Sınama Kavramları

- Sınama ve bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerin seçimi gerekmektedir.
- Bütünleştirme işleminde, en küçük birimlerden başlanarak sistem düzeyine çıkılmaktadır. Bu değişik düzeylere hitap edecek sınama yöntemleri olmalıdır.

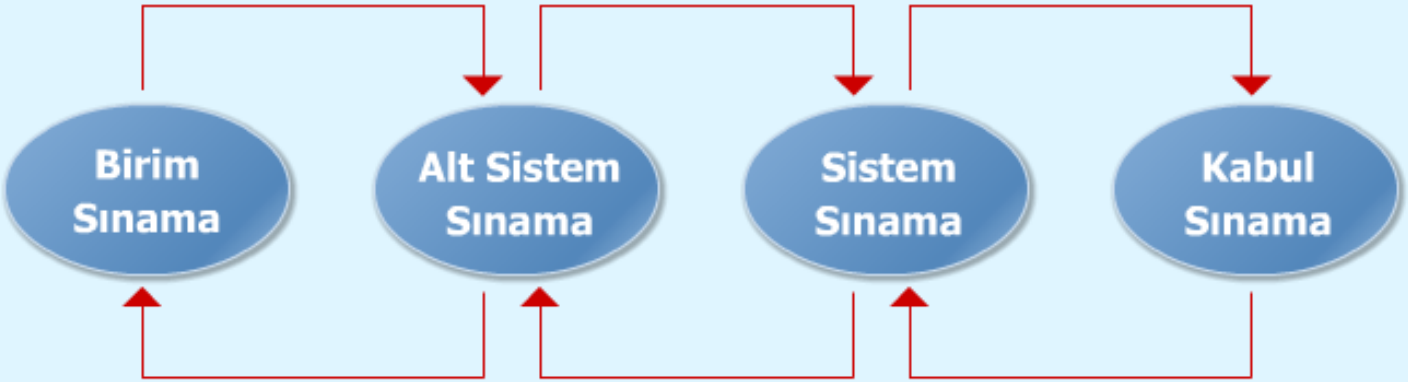
Sınama Kavramları



Birim Sınama

Bağlı oldukları diğer sistem unsurlarından bütünüyle soyutlanmış olarak birimlerin, doğru çalışmalarının belirlenmesi amacıyla yapılır. Birimler, ilişkili yapıtaşlarının bütünleştirilmesinden oluşurlar.

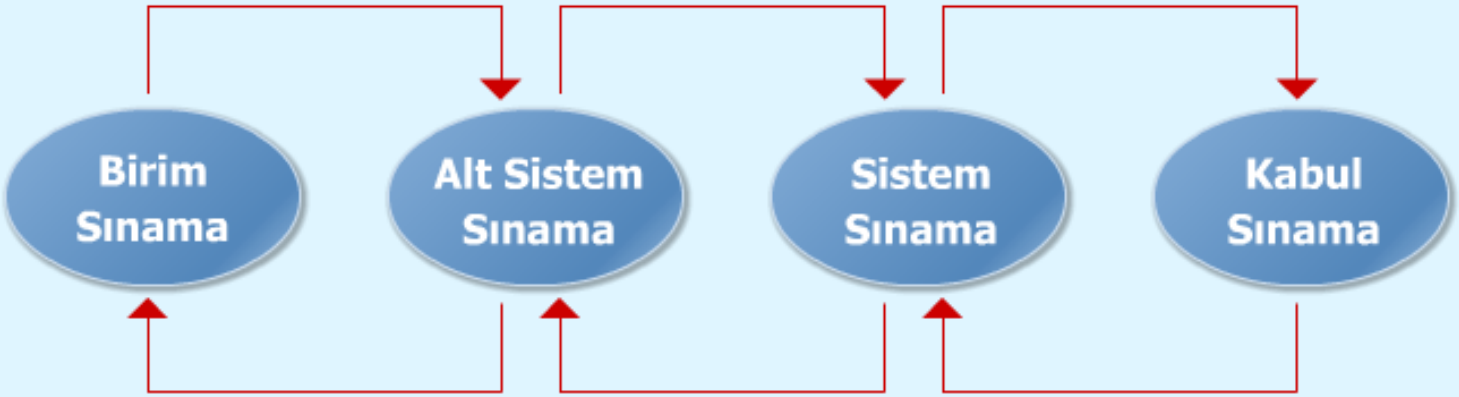
Sınama Kavramları



Alt-Sistem Sınama

Alt sistemler ise, modüllerin bütünleşmesi ile ortaya çıkar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu düzeyde en çok hata arayüzlerde bulunmaktadır, arayüz hatalarına yönelik sınamalara yoğunlaşılmalıdır.

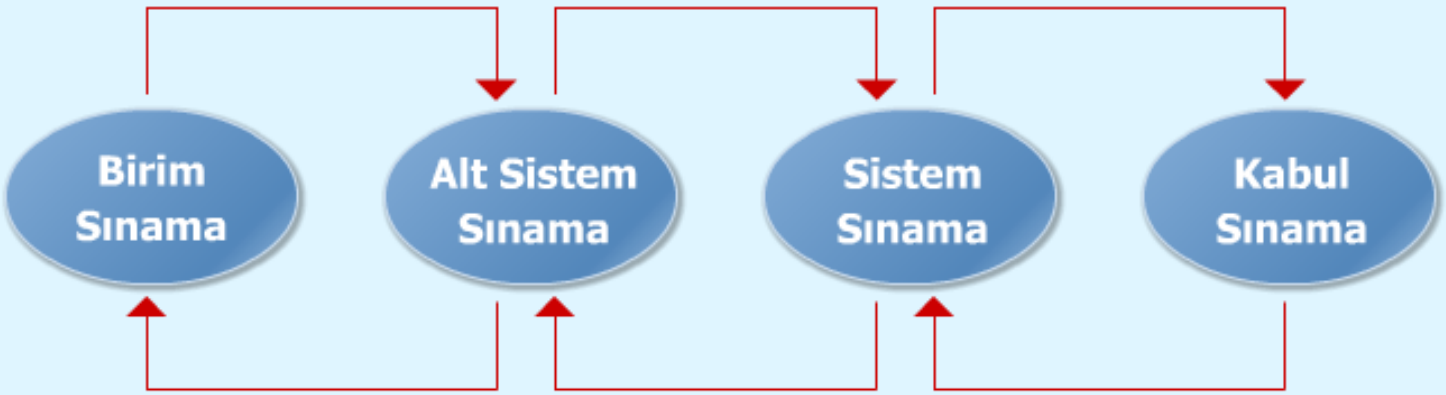
Sınama Kavramları



Sistem Sınaması

Üst düzeyde bileşenlerin sistem ile olan etkileşimlerinde çıkacak hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

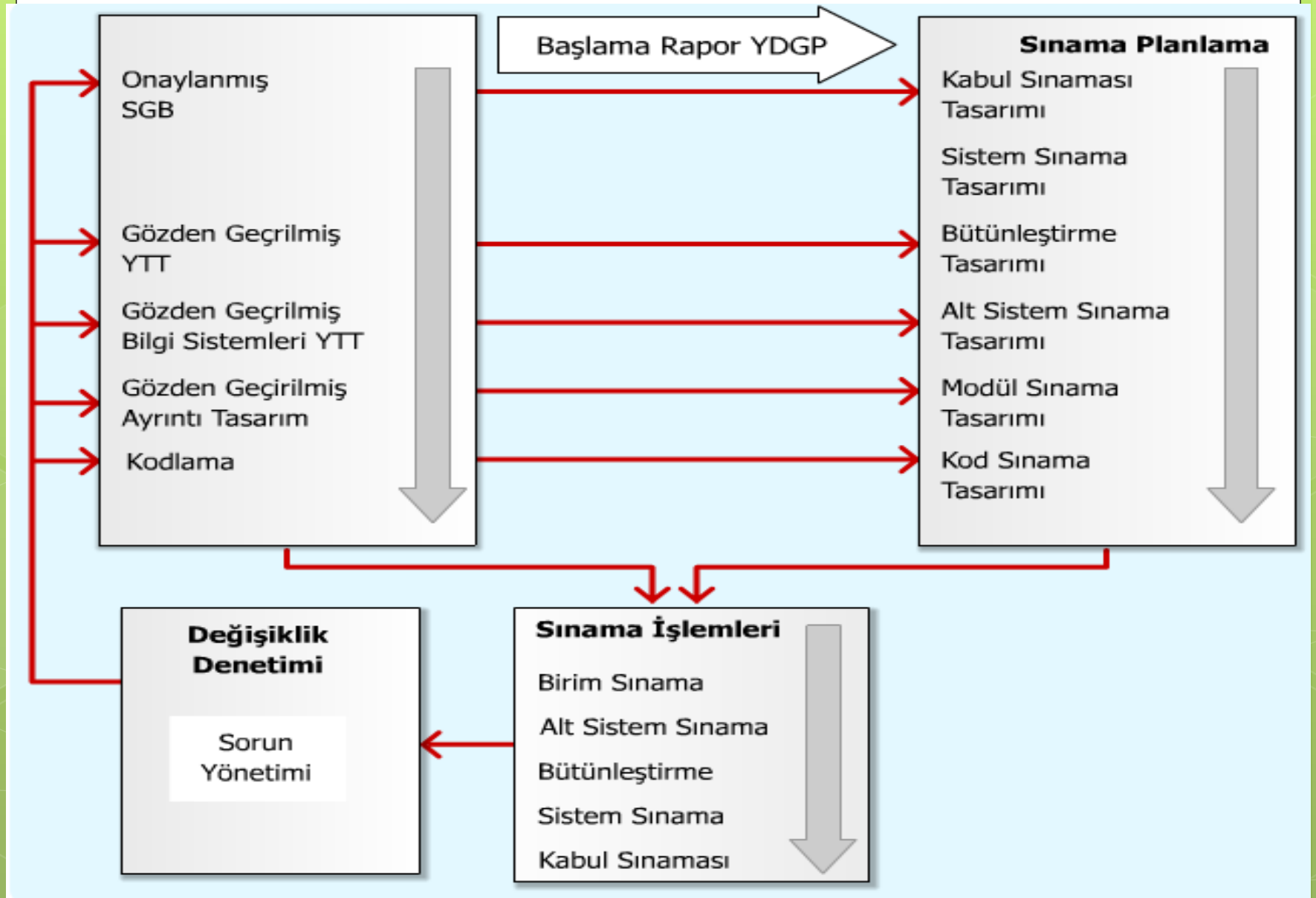
Sınama Kavramları



Kabul Sınaması

Çalıştırılmadan önce sistemin son sınamasıdır. Artık yapay veri yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması ve beta sınaması olarak ta bilinir. Alfa sınamada, tanımında, sınamanın geliştirici organizasyonun yerleşkesinde, kullanıcıların da gelerek katkıda bulunması içerilir. Daha sonra ürünün pazarlama işlemi sırasında beta sınama denilen, sınama, kullanıcının kendi yerleşkesinde, geliştirici gözetiminde yapılır.

Doğrulama ve Geçerleme Yaşam Döngüsü



Sınama Yöntemleri

- Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür.
- Her mühendislik ürünü, iki yoldan biri ile sınanır: Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün **(kara kutu - black box)** veya iç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanması **(beyaz kutu - white box)**.
- Kara kutu sınamasında sisteme, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır. Sonraki sayfalarda beyaz kutu sınamasına ilişkin bilgiler verilmektedir.

Beyaz Kutu Sınaması

- Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır.
- Yapılabilecek denetimler arasında:
  - Bütün bağımsız yolların en azından bir kere sınanması,
  - Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
  - Bütün döngülerin sınır değerlerinde sınanması,
  - İç veri yapılarının denenmesi bulunur.

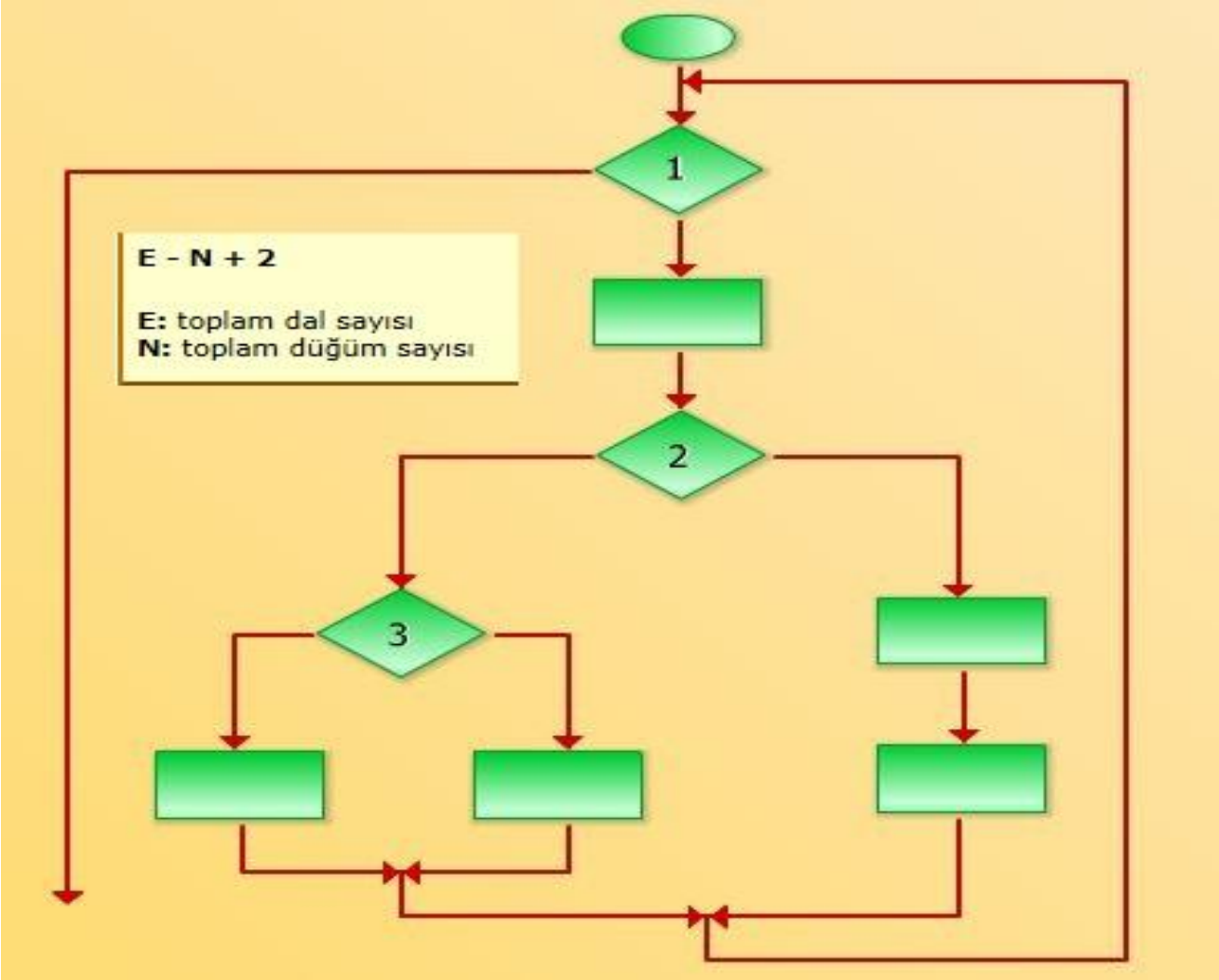
Beyaz Kutu Sınaması

- **Sınamaları yürütürken sınırlı çabamızı yerinde kullanmamız gerekir. Bunun için hataların bazı özelliklerinin bilinmesinde yarar vardır:**
- Bir program kesiminin uygulamada çalıştırılma olasılığı az ise o kesimde hata olması ve bu hatanın önemli olması olasılığı fazladır.
- Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları, aslında çok sıkça çalıştırılıyor olacaktır.
- Yazım hataları rasgele olarak dağılır. Bunlardan bazılarını derleyiciler bulur, bazıları da bulunmadan kalır.

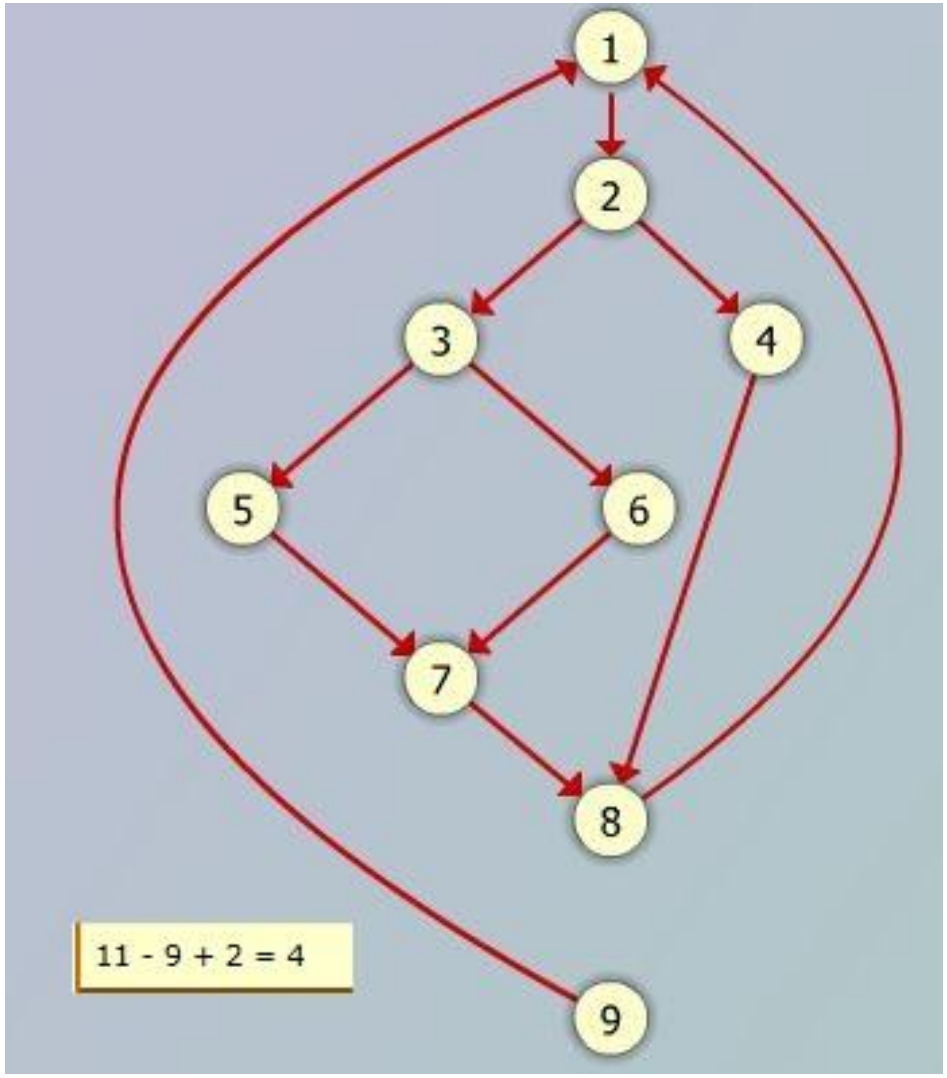
Temel Yollar Sınaması

- Daha önce çevrimsellik karmaşıklığı konusunda gördüğümüz hesap yöntemi ile bir programdaki bağımsız yollar bulunduktan sonra, bu kadar sayıda sınama yaparak programın her birimini bir şekilde sınamalara dahil etmiş oluruz.
- Bağımsız yolların saptanması için önce, program çizgesel bir biçime çevrilir.
- Bunu yapmak için ise, program iş akış şemaları diyagramları iyi bir başlangıç noktasıdır.

Temel Yollar Sınaması



Temel Yollar Sınaması



SINAMA ve BÜTÜNLEŞTİRME STRATEJİLERİ

- Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir.

- Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

Yukarıdan Aşağı Sınama ve Bütünleştirme

- Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur.

- En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir.

Yukarıdan Aşağı Sınama ve Bütünleştirme

- Ancak bu en üstteki bileşenin tam olarak sınanması için alttaki bileşenlerle olan bağlantılarının da çalışması gerekir.

- Alt bileşenler ise bu stratejiye göre henüz hazırlanmış olamazlar. Bunların yerine üst bileşenin sınaması için kullanılmak üzere 'koçan' programları yazılır.

Yukarıdan Aşağı Sınama ve Bütünleştirme

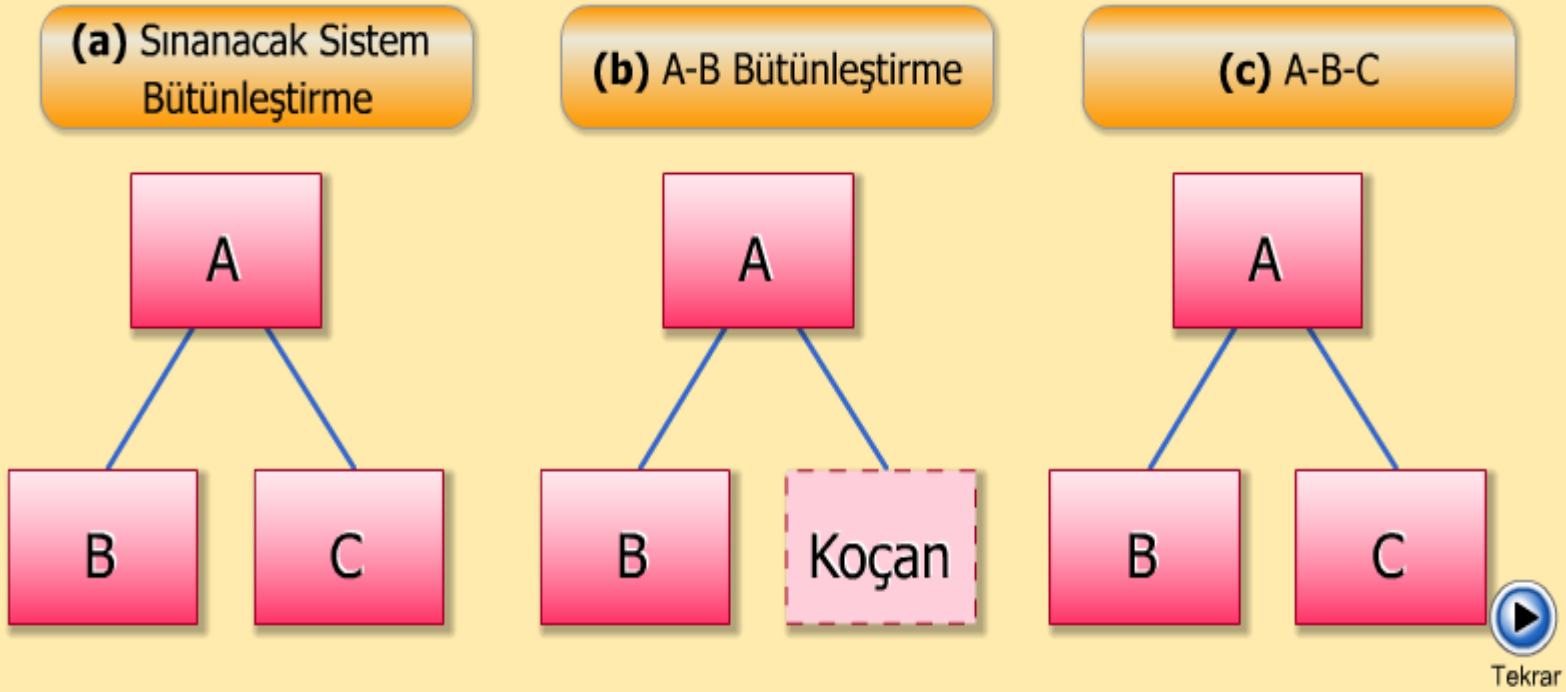
- Koçanlar, bir alt bileşenin, üst bileşen ile arayüzünü temin eden, fakat işlevsel olarak hiç bir şey yapmayan, boş çerçeve programlarıdır.

- Üst bileşenin sınanması bittikten sonra bu koçanlar, içleri doldurularak kendi kodlama ve birim sınama işlemlerini tamamladıktan sonra üst bileşen ile yeniden sınanırlar.

Yukarıdan Aşağı Sınama ve Bütünleştirme

Örnek:

A, B, C birimlerinden oluşan ve birim şeması şekil 7.5(a)'da belirtilen bir sistemin bu tür koçan kullanılarak sınanması şekil 7.5(b) ve şekil 7.5(c)'de belirtilmektedir.



Şekil 7.5: Bütünleştirme Sınamasında "koçan" Kullanımı

İlk adımda A ve B birimleri bütünleştirilir; C için bir "koçan" yazılır. İkinci adımda ise "koçan" kaldırılır ve C ile yer değiştirilerek A-B-C bütünleştirilir.

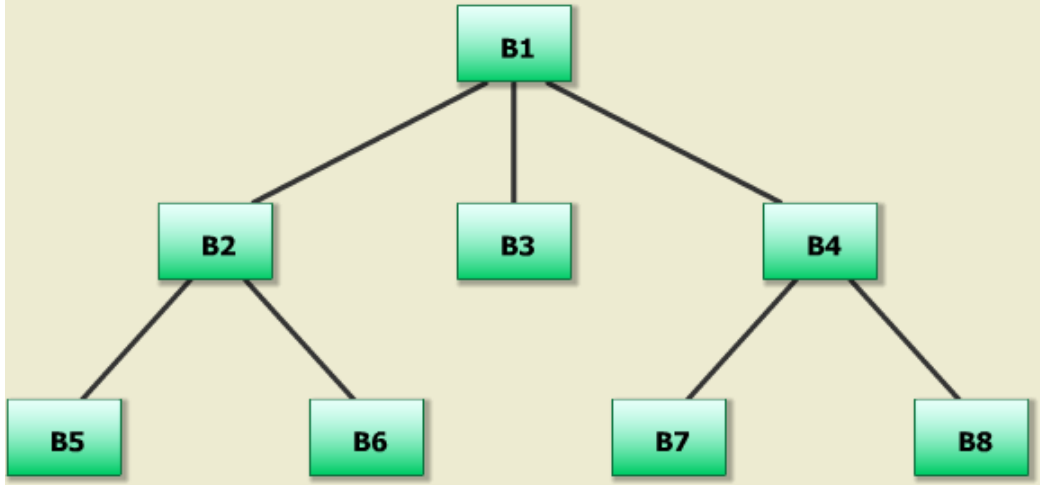
Yukarıdan Aşağı Sınama ve Bütünleştirme

Yukarıdan aşağıya doğru bütünleştirme işleminde iki yaklaşım izlenebilir:

- 1. Yaklaşım: Düzey Öncelikli Bütünleştirme
- 2. Yaklaşım: Derinlik Öncelikli Bütünleştirme

1. Yaklaşım: Düzey Öncelikli Bütünleştirme

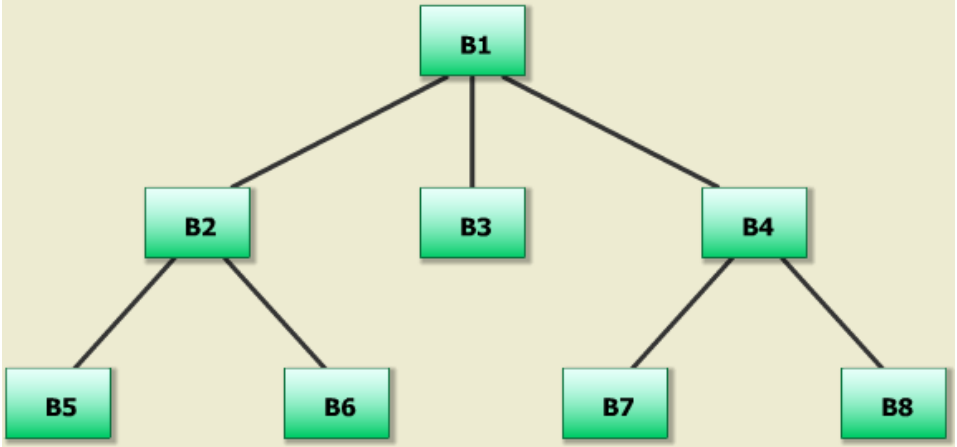
- En üst düzeyden başlanır, öncelikle aynı düzeylerdeki birimler bütünleştirilir.



1. Adım: B1-B2 (KoçanB3 - KoçanB4- KoçanB5 - KoçanB6)
2. Adım: B1-B2-B3 (KoçanB4- KoçanB5 - KoçanB6)
3. Adım: B1-B2-B3-B4 (KoçanB7 - KoçanB8- KoçanB5 - KoçanB6)
4. Adım: B1-B2-B3-B4-B5 (KoçanB7 - KoçanB8- KoçanB6)
5. Adım: B1-B2-B3-B4-B5-B6 (KoçanB7 - KoçanB8)
6. Adım: B1-B2-B3-B4-B5-B6-B7 (KoçanB8)
7. Adım: B1-B2-B3-B4-B5-B6-B7-B8

2. Yaklaşım: Derinlik Öncelikli Bütünleştirme

- En üst düzeyden başlanır. Birim şemasında bulunan her dal soldan sağa olma üzere ele alınır. Bir dala ilişkin bütünleştirme bitirildiğinde diğer dalın bütünleştirmesi başlar.



1. Adım: B1-B2 (KoçanB3 - KoçanB4- KoçanB5 - KoçanB6)
2. Adım: B1-B2 - B5 (KoçanB3- KoçanB4 - KoçanB6)
3. Adım: B1-B2-B5-B6 (KoçanB3 - KoçanB4)
4. Adım: B1-B2-B5-B6-B3 (KoçanB4)
5. Adım: B1-B2-B5-B6-B3-B4 (KoçanB7 - KoçanB8)
6. Adım: B1-B2-B5-B6-B3-B4-B7 (KoçanB8)
7. Adım: B1-B2-B5-B6-B3-B4-B7-B8

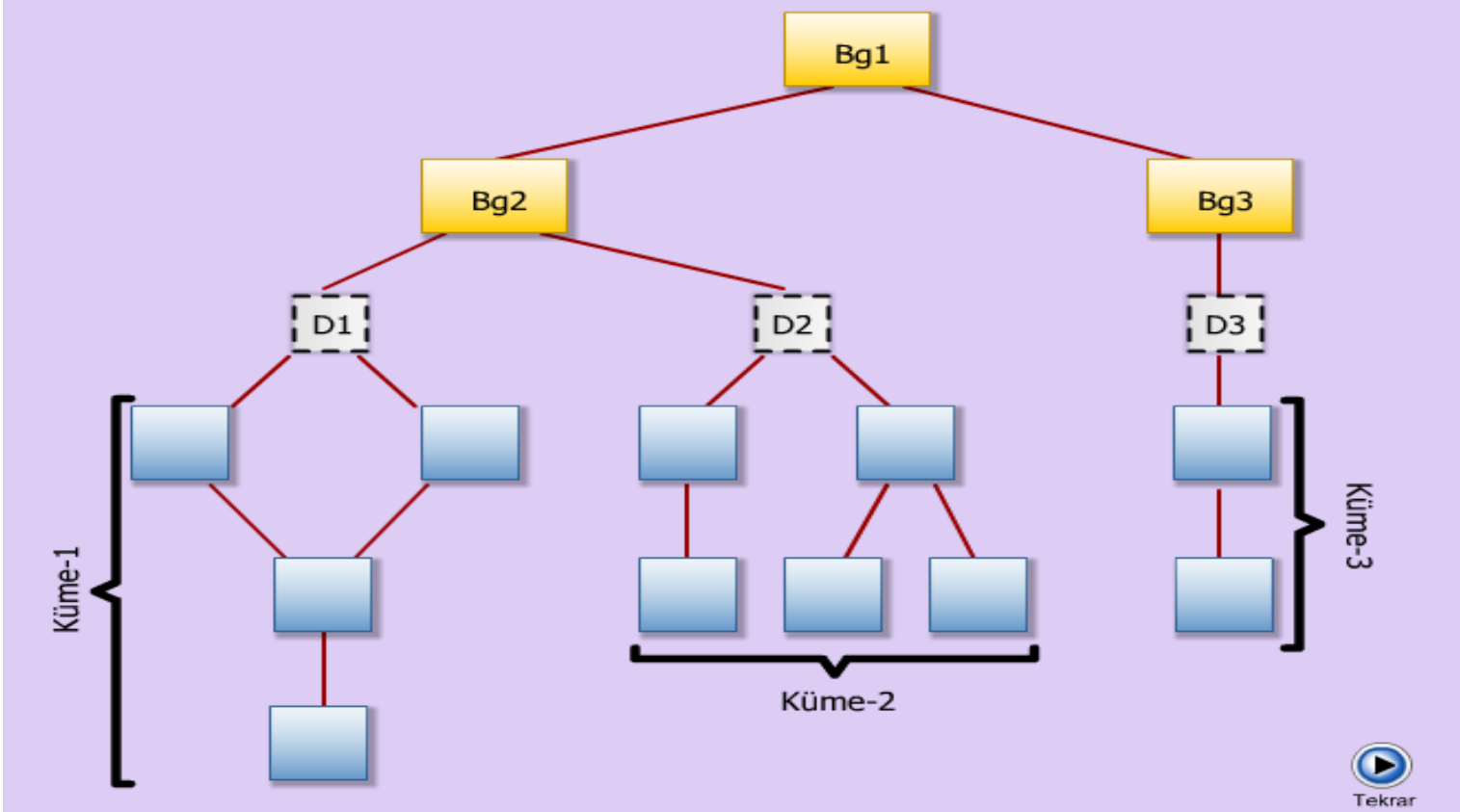
Aşağıdan Yukarıya Sınama ve Bütünleştirme

- Önce en alt düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınanması gerektiğinde bu üst bileşen, bir '**sürücü**' ile temsil edilir.
- Yine amaç, çalışmasa bile arayüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir.

Aşağıdan Yukarıya Sınama ve Bütünleştirme

- Bu kez kodlama, bütünleştirme ve sınama aşağı düzeylerden yukarı düzeylere doğru gelişir ve yukarı düzeylerde önce sürücü olarak yazılan birimler sonra gerçekleriyle yer değiştirerek o düzeyin birimleri/alt sistemleri olurlar.

Aşağıdan Yukarıya Sınama ve Bütünleştirme



Şekil 7.7: Aşağıdan Yukarı Bütünleştirme

Şekilde;

1. Belirli bir yazılım alt işlevini gören alt düzey birimler **kümeler** biçiminde oluşturulurlar,
2. Denetim amaçlı bir **sürücü** programı sınama işlemi için girdi ve çıktı oluşturmak amacıyla yazılır,
3. Sürücüler aşağıdan yukarı kaldırılır ve gerçek birim ya da birim kümeleriyle değiştirilerek sınama işlemi sürdürülür.

SINAMA PLANLAMASI

- Sınama işlemi çok kapsamlıdır. Bir plan güdümünde gerçekleştirilmelidir. Böyle bir planın temel bileşenleri önceki sayfalarda belirtilmiştir.
- Yazılım yaşam döngüsünün süreçlerine koşut olarak, farklı ayrıntı düzeylerinde birden fazla sınama planı hazırlanır.

Giriş
Amaç Tanım ve Kısaltmalar Referanslar

SINAMA PLANLAMASI

Sınama Yönetimi

Sınama Konusu
Sınama Etkinlikleri ve Zamanlama

Temel Sınama Etkinlikleri
Destek Etkinlikler

Kaynaklar ve Sorumluluklar
Personel ve Eğitim Gereksemeleri
Sınama Yaklaşımı
Riskler ve Çözümler
Onaylar

SINAMA PLANLAMASI

Sınama Ayrıntıları

- Sınanacak Sistemler
- Girdiler ve Çıktılar
- Sınamaya Başlanma Koşulları
  - Girdilerin Hazır Olması
  - Ortam Koşulları
  - Kaynak Koşulları
- Sınama Tamamlama Kıstası
- Sınama Geçme-Kalma Kıstası
- Sınama Askıya Alınma Kıstası ve Sürdürme Gerekleri
- Sınama Sonuçları

- Sınama planları; Birim (Modül) Sınama Planı, Alt Sistem Sınama Planları, Bütünleştirme Sınama Planları, Kabul Sınama Planları, Sistem Sınama Planları biçimindedir.

SINAMA BELİRTİMLERİ

- Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir. Bu ayrıtılar temel olarak:
- sınanan program modülü ya da modüllerinin adları,
- sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- sınama verileri,
- sınama senaryoları

türündeki bilgileri içerir.

SINAMA BELİRTİMLERİ

- Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

- Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmasıdır.

SINAMA BELİRTİMLERİ

- Sınama işlemi sonrasında bu belirtimlere,

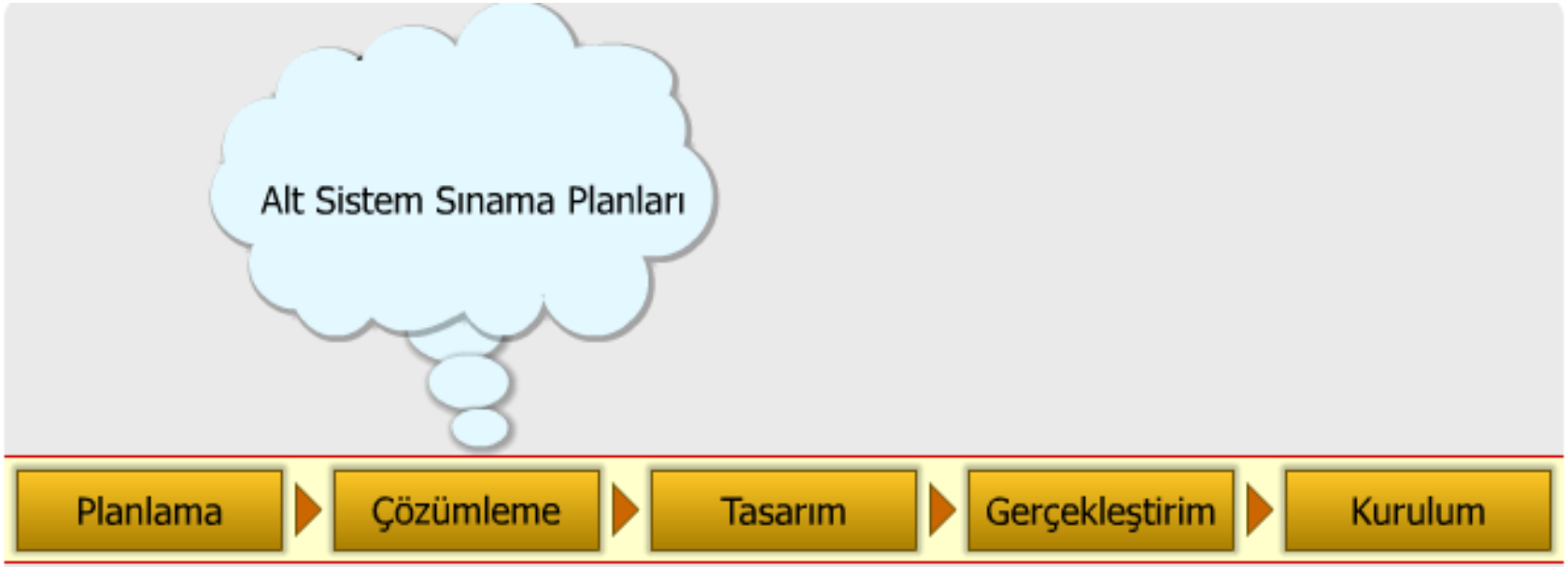
- sınamayı yapan,
- sınama tarihi,
- bulunan hatalar ve açıklamaları

türündeki bilgiler eklenerek sınama raporları oluşturulur.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

Planlama aşamasında genel sınama planı oluşturulur. Söz konusu plan tüm sınama etkinliklerini çok genel hatlarıyla tanımlar ve sınama planlamasında verilen bilgileri içerir.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



Çözümleme aşamasında, sistemler ve alt sistemler ortaya çıkarılır ve sınama planı alt sistemler bazında ayrıntılandırılır.

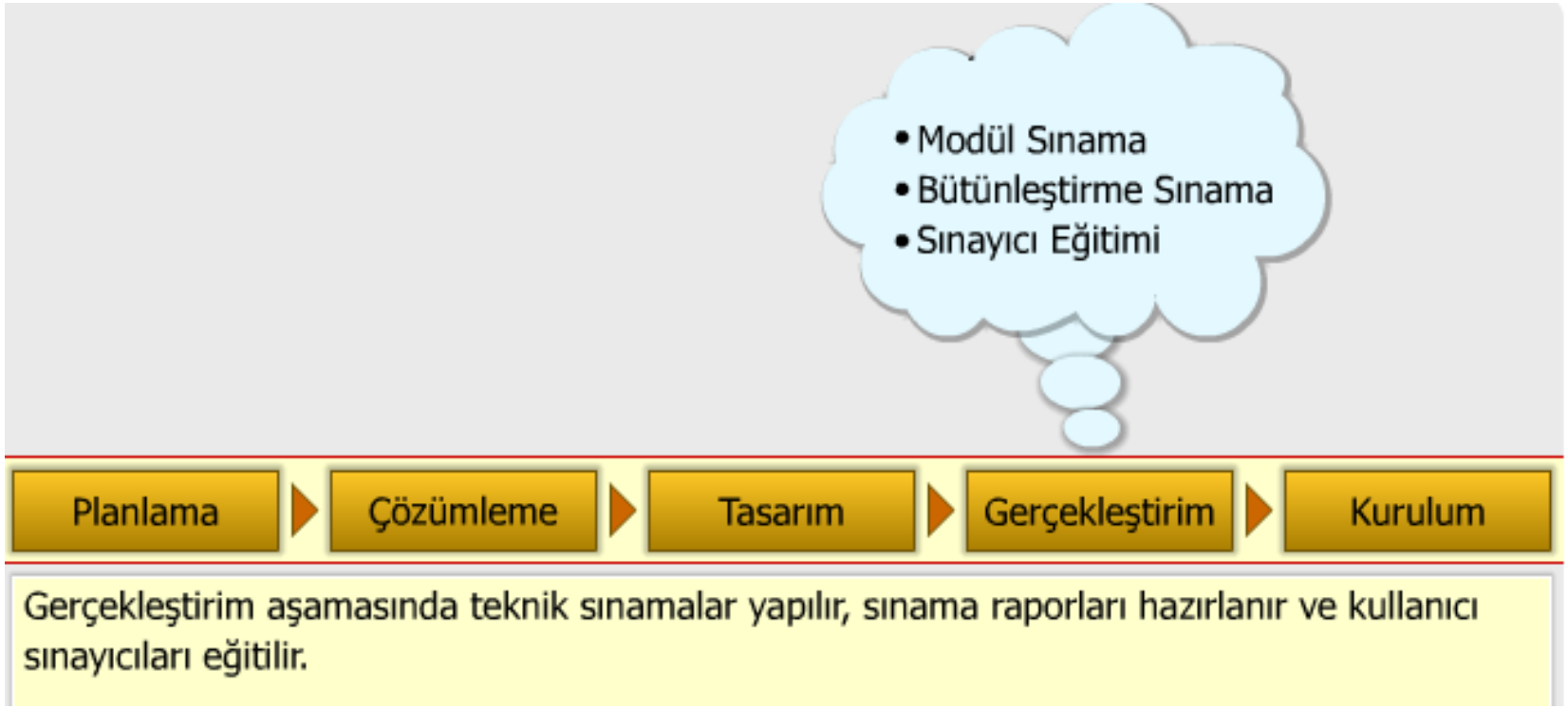
YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- Modül Sınama Planı
- Sınama Belirtimleri
- Sınama Eğitim Kılavuzları

Planlama ▶ Çözümleme ▶ Tasarım ▶ Gerçekleştirim ▶ Kurulum

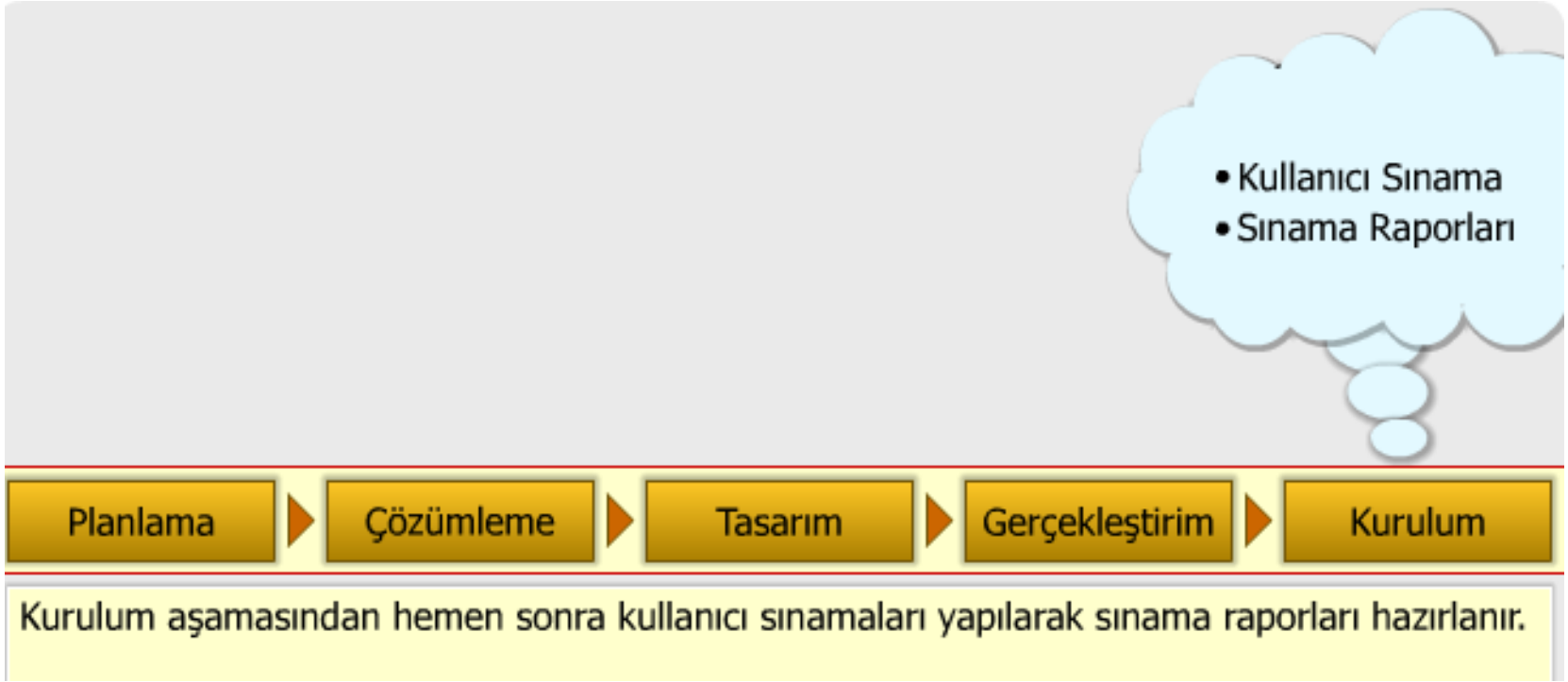
Tasarım aşaması, tüm yazılım modüllerinin ortaya çıkarıldığı aşamadır. Bu aşamanın başlangıcında yazılım modülleri için sınama planı detaylandırılır ve sınama belirtimleri hazırlanır. Söz konusu belirtimler, kullanıcı kılavuzları ile birlikte sınamaya ilişkin eğitim için temel bilgileri oluşturur.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



Gerçekleştirim aşamasında teknik sınamalar yapılır, sınama raporları hazırlanır ve kullanıcı sınayıcıları eğitilir.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- Sınama sırasında bulunan her hata için, değişiklik kontrol sistemine (DKS), "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

Onulmaz Hatalar: BT projesinin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- **Küçük Hatalar:** Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.

- **Şekilsel Hatalar:** Heceleme hatası gibi önemsiz hatalardır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Bu kısımda, gerçek yaşam ortamında, **Oracle Designer CASE** aracı ve **Developer** görsel yazılım geliştirme platformu kullanılarak geliştirilen yazılım modüllerinin sınanması işleminin nasıl yapılacağı ve buraya kadar açıklanan sınama yöntemlerinin nasıl uygulandıkları bir örnek üzerinde anlatılmaktadır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Oracle Developer kullanılarak geliştirilen her yazılım **form**lardan oluşur.
- Bir form, bir ekran ve bu ekranda yapılan işlemlere karşılık gelen PL/SQL kodları biçiminde tanımlanır.
- Bu örnekte elimizde, sınama işlemine koşulacak ve uygulamanın çeşitli işlevlerine ilişkin bir dizi form olduğunu düşünebiliriz.

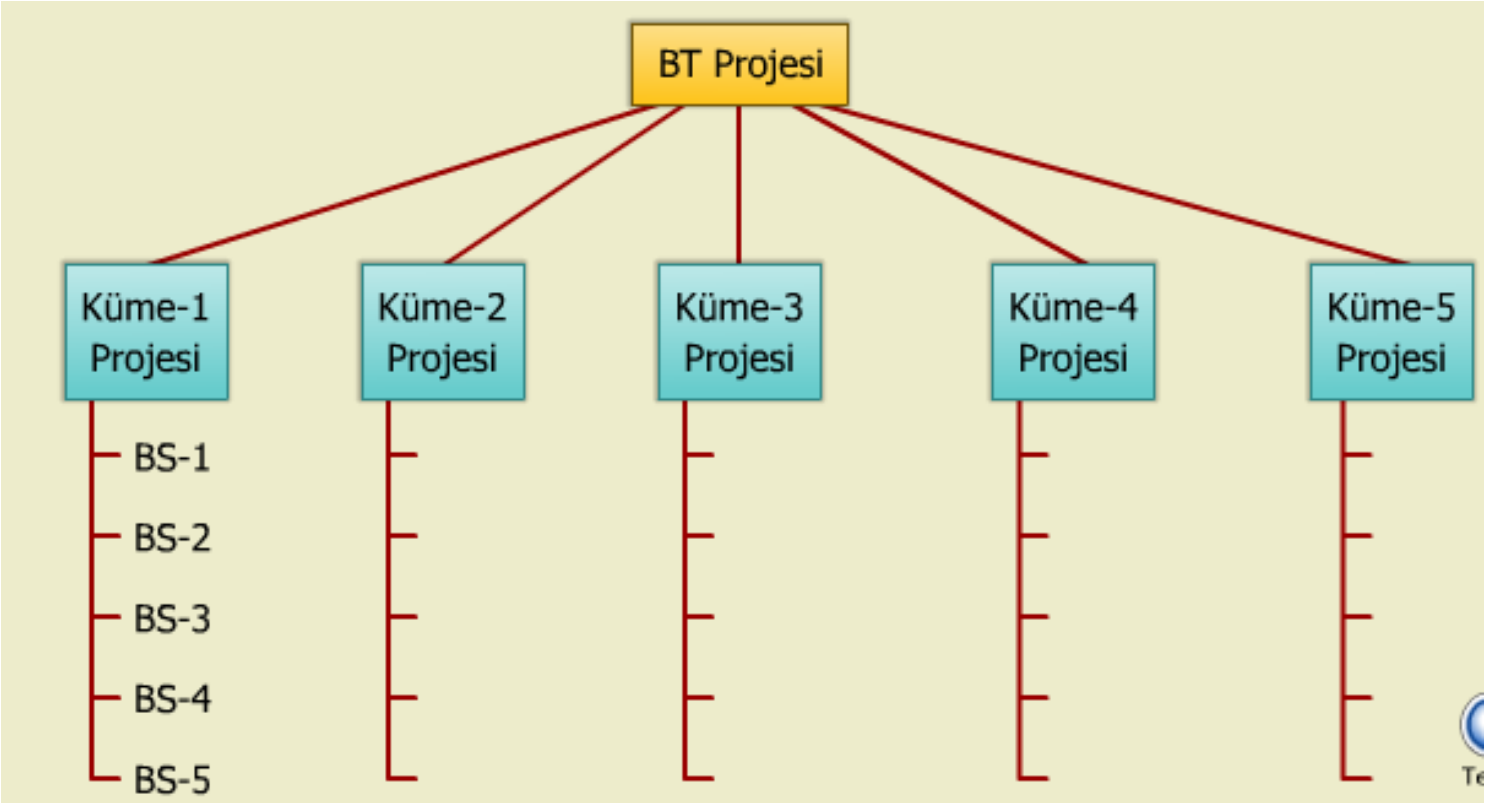
BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Bu örnekte söz edilen uygulama, 2000'den fazla kullanıcısı olan, ülkenin çeşitli yörelerine dağılmış birimlerde çalışacak biçimde tasarlanmış ve 1000'den fazla Developer formundan oluşmaktadır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Uygulamanın sınama aşamasına gelmesi, 2 yıllık bir süre ve yaklaşık 100 kişi-yıl'lık bir iş gücü gerektirmiştir.
- Uygulama beş ana kümeye bölünmüş ve her küme belirli sayıda bilgi sistemini içermektedir.
- Toplam olarak 30 bilgi sistemi bulunmaktadır. Uygulama sıra düzeni Şekil 7.10'da verilmektedir.

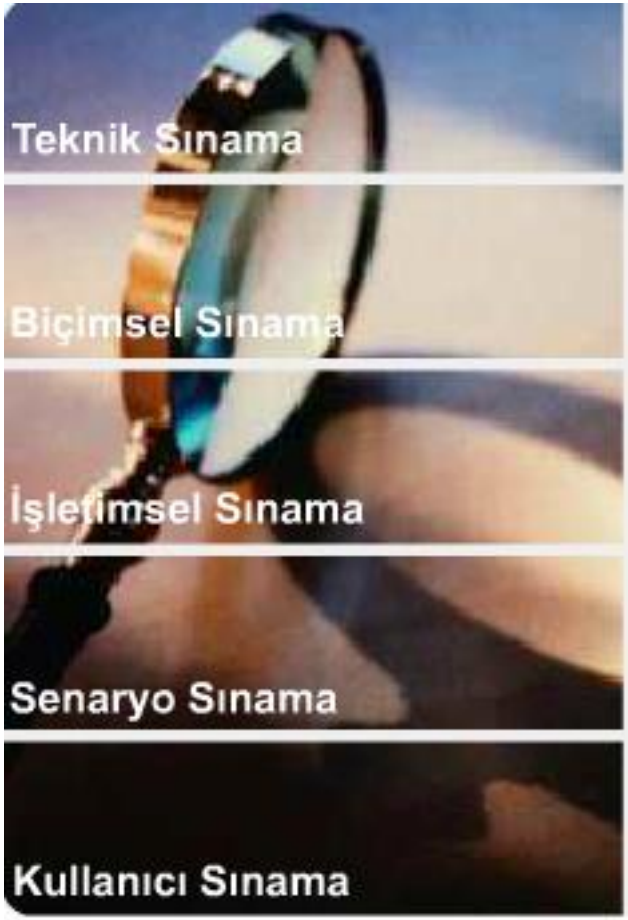
BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama



Sınama Ortamı Oluşturulması

- Üretimin etkilenmemesi amacıyla, yalnızca sınayıcıların kullanacakları ve ayrı bilgisayarlardan oluşan bir sınama ortamı oluşturuldu.
- Oluşturulan sınama ortamı ile üretim ortamının birebir aynı olması sağlandı.
- Üretimi biten yazılım parçaları, bir kayıt düzeni içerisinde sınama ortamına alındı.

Sınama Yöntemlerine Karar Verilmesi

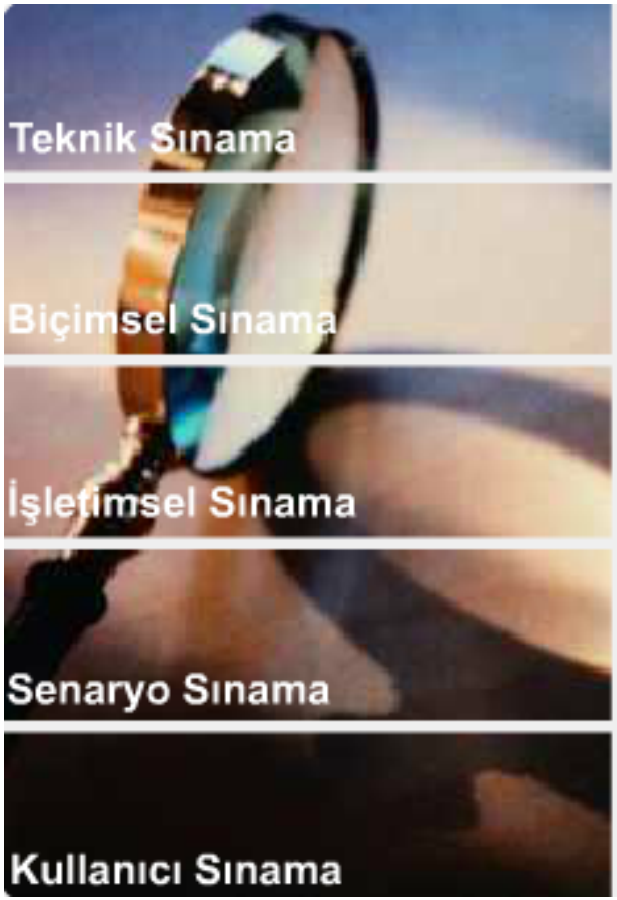


Teknik Sınama

Üretim ortamında yapılacak sınama olarak karar verildi. Bu sınama, modül sınaması ve bütünleştirme sınaması olarak üretim ekipleri tarafından gerçekleştirildi. Modül sınama yöntemi olarak "beyaz kutu" sınama yöntemi uygulandı. Tüm program deyimleri en az bir kez, tüm döngüler en az 10 kez yinelenecek biçimde sınama yapıldı.

Bütünleştirme sınama yöntemi olarak, "yukarıdan aşağıya sınama yöntemi" ve "derinlik öncelikli bütünleştirme" stratejisi uygulandı.

Sınama Yöntemlerine Karar Verilmesi



Biçimsel Sınama

Üretim ekiplerinden bağımsız olarak, sınama ekipleri tarafından yapılan sınamadır. Bu sınama, Developer formları üzerinde görsel olarak yapıldı. Amaç, formların, önceden kararlaştırılan standartlara uygunluğunun saptanmasıydı. Örneğin, form alanları, kararlaştırılan uzunlukta mı? Başlıklar istenilen gibi koyu mu? Yardım düğmesi hep aynı yerde mi vb.

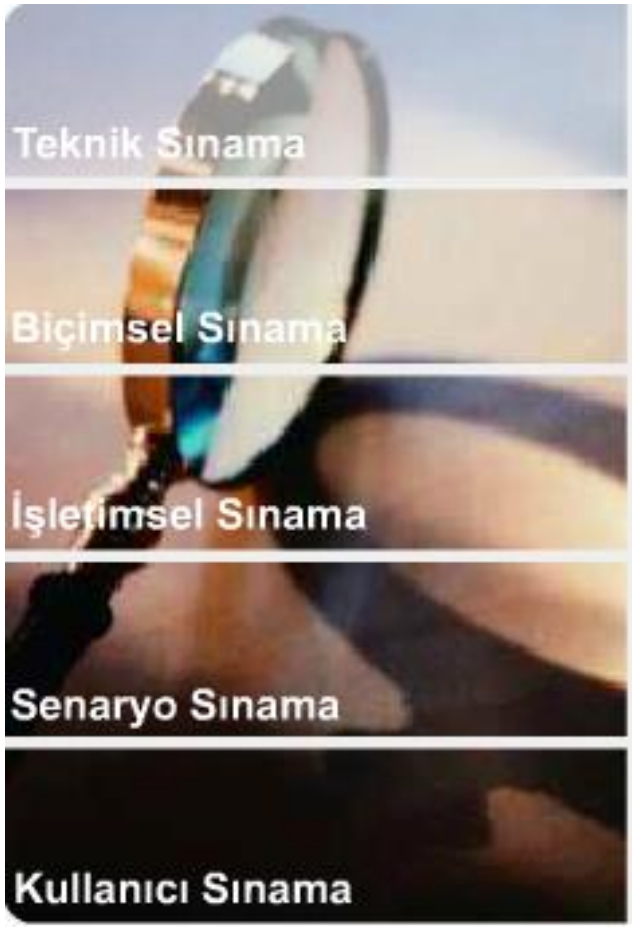
Sınama, formlar işletilmeden yapılır. Tüm formlar tek tek incelenir ve standartlara uygun olmayanlar belirlenip, düzeltilmek üzere üretim ekibine geri iletilir. Biçimsel sınamaların yapılması amacıyla denetim listeleri hazırlanır ve sınama sırasında bu listeler kullanılır. Listelere kaydedilen her sonuç DKS'ye aktarılır. Bu yolla üretim ekiplerinin performansı izlenebilir.

Sınama Yöntemlerine Karar Verilmesi

İşletimsel Sınama

Üretim ekiplerinden bağımsız olarak, sınama ekipleri tarafından yapılan sınamadır. Biçimsel sınama işlemi bittikten sonra yapılır. Bu sınamada her form ayrı ayrı çalıştırılarak işlem yapılır. Amaç, formun çalışıp çalışmadığının belirlenmesidir. Form alanlarının sınır değerlerle çalışıp çalışmadığı, aykırı değer verildiğinde uygun hata iletisi alınıp alınmadığı vb belirlenmeye çalışılır.

Sınama Yöntemlerine Karar Verilmesi

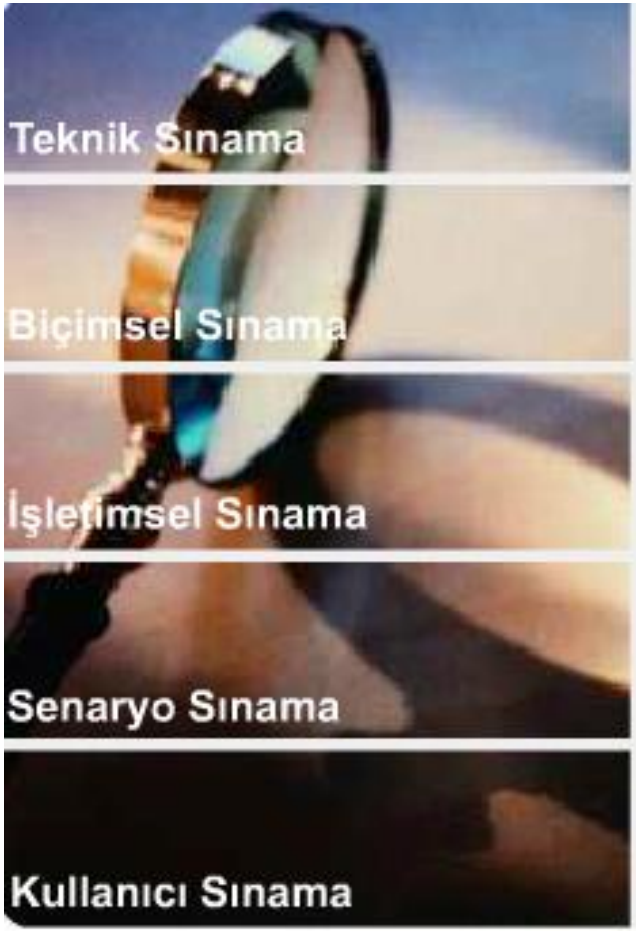


Senaryo Sınaması

Sınama ekipleri tarafından yapılan sınamadır. Ancak, senaryoların hazırlanması sırasında üretim ekipleri ile birlikte çalışılır.

Amaç, birden fazla formun bir arada sınanmasıdır. Bu amaçla, “senaryo”lar hazırlanır. Her senaryo, çözümleme aşamasında belirlenen bir iş fonksiyonuna karşılık gelecek biçimde. hazırlanır.

Sınama Yöntemlerine Karar Verilmesi



Kullanıcı Sınaması

Kullanıcılar tarafından yapılması öngörülen sınamadır. Senaryo sınamasının kullanıcı tarafında yapılan biçimi olarak düşünülebilir.

Kullanıcı Sınama Eğitimi

- Sınama yapılacak kullanıcı sınayıcılarına, sınamaların nasıl yapılacağına ilişkin eğitim verilmesi gerekmektedir.
- Eğitim kitapçıklarının hazırlanması amacıyla, senaryo sınamalarında kullanılan "senaryo"lar ve kullanıcı kitapçıkları kullanılır.

Sınamaların Yapılması

Sınamalar sırasıyla:

- Teknik Sınama,
- Biçimsel Sınama,
- İşletimsel Sınama,
- Senaryo Sınaması,
- Kullanıcı Sınama

biçiminde yapılır.

Alınan Dersler 1

- Bir taraftan üretimin yapıldığı, öte yandan ise kullanıcı tarafında sınamaların yapıldığı bir ortamda, üretim ekipleri bir yandan yeni yazılım parçaları geliştirme, öte yandan ise sınama sonucu bildirilen hataları düzeltme durumu ile karşı karşıyadır.

Alınan Dersler 1

- Bu durum, zaman zaman üretim ekiplerinde dirençlere neden olmaktadır.
- Büyük projeler için, kaçınılmaz olan bu tür durumların iyi izlenmesi ve planlanması gerekmektedir.
- Bu anlamda Kalite ekibi oldukça önem kazanmaktadır.

Alınan Dersler 2

- Sınama yapılacak ortam ile üretim ortamının, fiziksel olarak birbirinden ayrı olarak düzenlenmesi çok önemidir.
- Aksi durumda, sınama sırasında, sistemlerin kilitlenmesi ve veri tabanının zarar görmesi vb. sorunlarla karşılaşılır.
- Yapılan işlerin izlenmesi zorlaşır.

Alınan Dersler 3

- Kullanıcı sınayıcı eğitimlerinin zaman zaman yinelenmesi gerekir.
- Kurumlardaki eleman değişiminin fazla olması sonucu, eğitim almamış kullanıcıların sistemi sınaması gibi durumlarla karşılaşılır ki bu da projeyi olumsuz olarak etkiler.

Alınan Dersler 4

- Kullanıcı sınayıcıları, kendi işlerinin yoğunluğunu öne sürerek, sınama işlemine gereken önemi gösterememektedir.
- Bu durumda Yerinde Destek ekiplerine önemli görevler düşmektedir.

Sorular

1. Doğrulama ile Geçerleme arasındaki farklılıkları belirtiniz. Birer örnekle açıklayınız.

2. Sınama Yöntemlerini açıklayınız.

3. "Beyaz Kutu" sınama ile "Temel Yollar Sınama" yöntemleri arasındaki farlılıkları belirtiniz.

4. Sınama Yöntemleri ile sınama belirtimleri arasındaki farkı belirtiniz.

5. Yukarıdan aşağıya doğru bütünleştirme ve aşağıdan yukarıya bütünleştirme yöntemlerinin zorluklarını ve kolaylıklarını belirtiniz.

6. Sınama belirtimlerinin önemi nedir.

7. Kullanıcı sınaması sırasında yaşanabilecek sorunları belirtiniz.

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

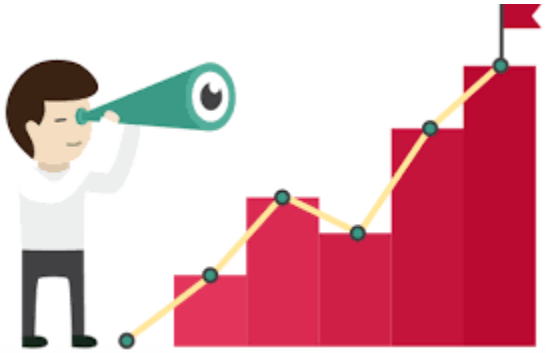
  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ
9.Hafta

Bakım

Bölüm Hedefi

- Geliştirilen yazılımın uygulamaya alınabilmesi için gerekli yöntemler ve yazılımın çalışması sırasında yapılması gereken bakım işlemleri bu bölümde açıklanmaktadır. Bölümün temel hedefleri arasında, IEEE 1219 standardının tanıtımı bulunmaktadır.



Giriş

- Sınama işlemleri bitirilen yazılımın kullanıcı alanına yüklenmesi ve uygulamanın başlatılması gerekmektedir.
- Yazılım kullanıma geçtikten sonra, yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan "bakım" aşaması başlar.
- İzleyen kesimlerde, kurulum ve bakım aşamasında yapılması gerekenler açıklanmaktadır.
- Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı dikkate alınmıştır.

KURULUM

- Sınanmış yazılımların kullanıcı sahasına aktarılması ve yazılımın gerçek yaşamda uygulamasının başlatılması için yapılan işlemler kurulum işlemleri olarak tanımlanmaktadır.

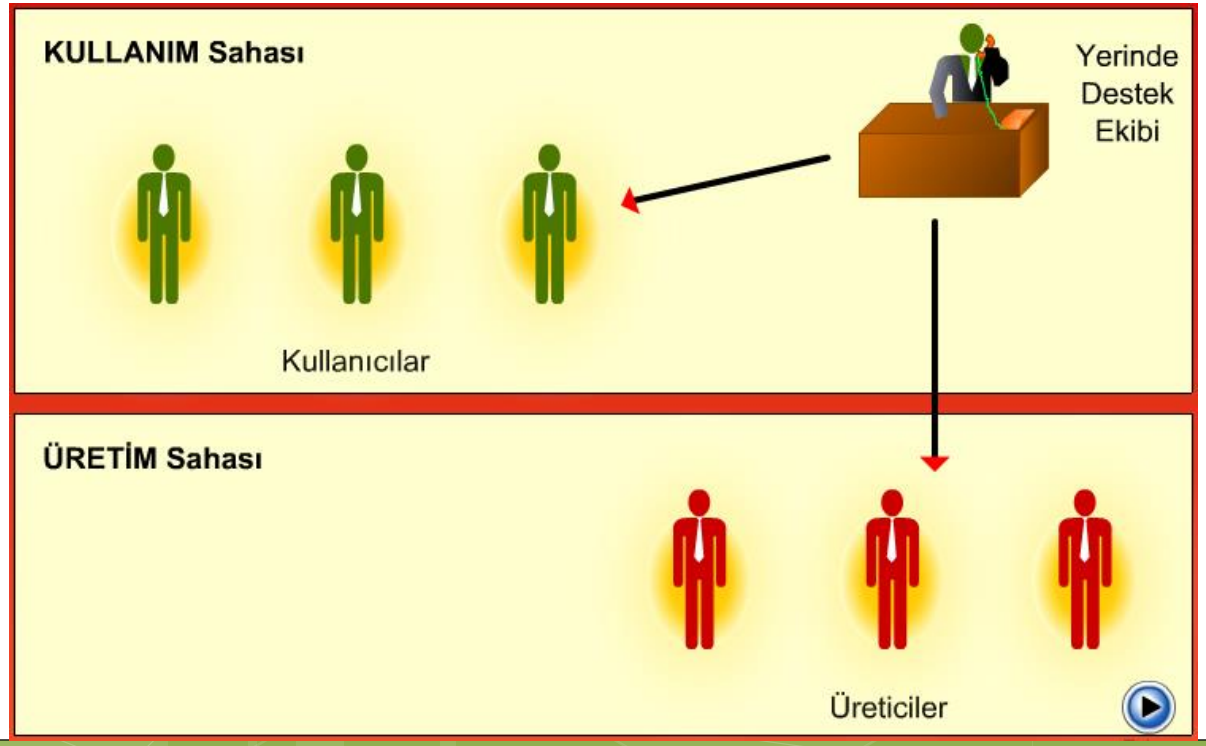
Yerinde Destek Organizasyonu

- Yerinde destek ekibi, kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir. Bu ekibin temel görevleri:

- Kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak,
- Giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasındaki uygulama yazılımı destek ekibine iletmek,

Yerinde Destek Organizasyonu

- Kullanıcıya iş başında uygulama eğitimi vermek,
- Kullanıcı sınama günlüklerini toplamak
- Yapılan tüm işlemleri konfigürasyon veri tabanına kaydetmek biçimindedir.



YAZILIM BAKIMI

1. Tanım
2. Bakım Süreç Modeli
   2.1. Sorun Tanımlama Süreci
   2.2. Çözümleme Süreci
   2.3. Tasarım Süreci
   2.4. Gerçekleştirim Süreci
   2.5. Sistem Sınama Süreci
   2.6. Kabul Sınaması Süreci
   2.7. Kurulum Süreci

başlıkları altında incelenecektir.

1.Tanım

- Bakım, işletime alınan yazılımın sağlıklı olarak çalışması ve ayakta kalabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlanır. Uygulamada çalışan bir yazılımın üç tür bakım gereksinimi bulunmaktadır:
  - Düzeltici Bakım
  - Uyarlayıcı Bakım
  - En İyileyici Bakım

1.Tanım

Düzeltici Bakım

Teorik olarak bir yazılımın tümüyle sınanabilmesi olası olsa bile, pratikte bu sağlanamaz. Bu nedenle çalışan bir yazılımda her an hata ile karşılaşma olasılığı vardır. Bu nedenle zaman zaman çalışan yazılımda ortaya çıkan hataların düzeltilmesi gerekir. Bu tür düzeltme çalışmaları "Düzeltici bakım olarak adlandırılır.

1.Tanım

Uyarlayıcı Bakım

Uygulama yazılımları, işletme ya da kuruluşlarn günlük yaşamlarında yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan araçlardır. Her kuruluş ya da işletme canli bir varlık gibi düşünülebilir. Hiçbir işletme durağan değildir. Süreç içinde değişkenlik gösterir. İşletme ya da kuruluşlarda değişiklik, yapılan işlerin yapılma tarzının değişmesi, yeni iş türlerinin ortaya çıkması biçiminde kendini gösterebilir. İşletme ya da kuruluşlarda yaşanan değişikliklerin, o kuruluşun işlerini bilgisayar yardımı ile yapmalarını sağlayan uygulama yazılımlarına da yansıtılması gerekir. Bu yansıtma işlemi, "Uyarlayıcı Bakım" olarak tanımlanır.

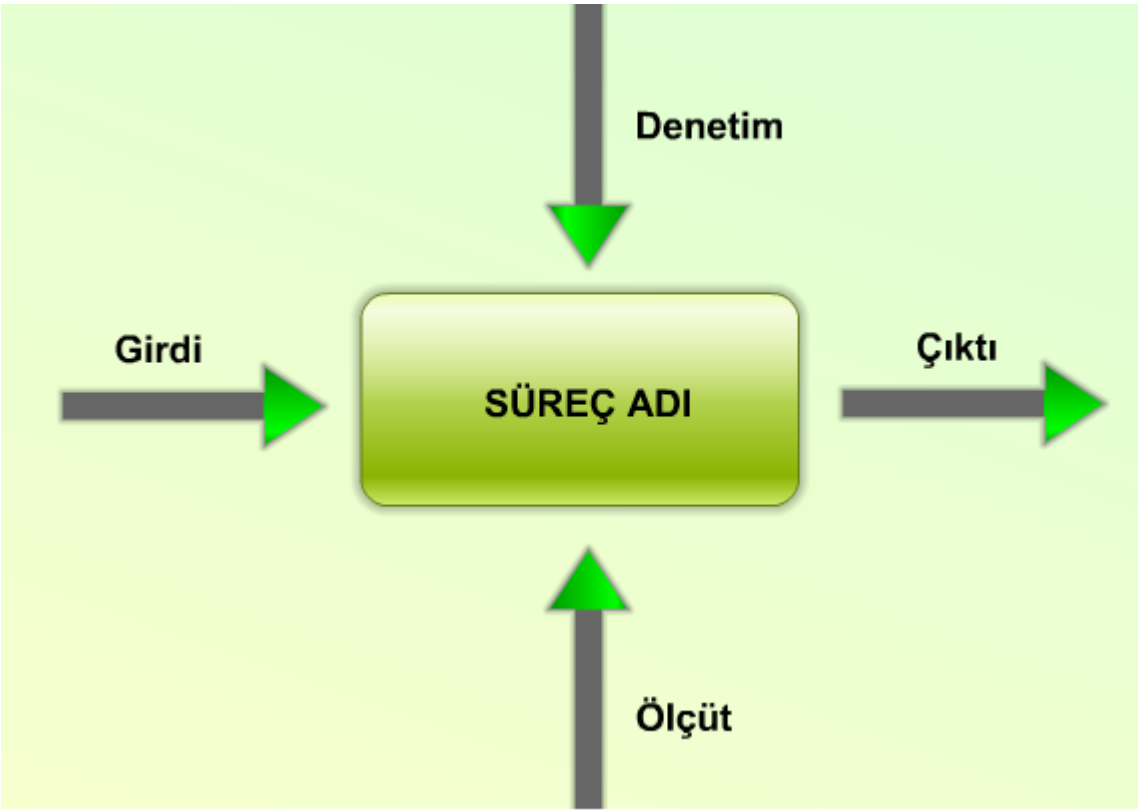
1.Tanım

En İyileyici Bakım

Zaman zaman uygulama yazılımlarının çalışma performanslarının iyileştirilmesi amacıyla yapılan çalışmalar "en iyileyici bakım" olarak tanımlanır.

2. Bakım Süreç Modeli

- IEEE 1219 standardı tarafından önerilen bakım süreç modeli şekilde belirtilen şablonu kullanarak bakım süreçlerini tanımlamaktadır.



2. Bakım Süreç Modeli

Bakım süreç modelinin süreçleri:

- **Sorun Tanımlama/sınıflandırma,**
- **Çözümleme,**
- **Tasarım,**
- **Gerçekleştirim,**
- **Sistem Sınama,**
- **Kabul Sunama**
- **Kurulum** biçimindedir. Görüldüğü gibi, bakım süreci, yazılım yaşam döngüsü çekirdek adımlarının bir anlamda yinelenmesinden oluşmaktadır. Bu yinelenme yalnızca değişiklik isteklerinin var olan koda aktarılması amacıyla yapılmaktadır. Bakım süreçleri aşağıda açıklanmaktadır.

2.1. Sorun Tanımlama Süreci

2.1. Sorun Tanımlama Süreci

Girdi
Sorun tanımlama sürecinin temel girdisi, Bakım isteğidir. Herhangi bir bakım isteğine örnek olarak: Sistemde beklenen ve yeni düzenlemelere ilişkin değişiklikler, • Yeni işlev istekleri, • Yazılımda bulunan yanlışların düzeltilme istekleri, • Performans artımına ilişkin istekler, • Yeni iş yapma türlerine ilişkin istekler ve • Teknolojinin zorlaması sonucu oluşan istekler türündeki istekler verilebilir.

2.1. Sorun Tanımlama Süreci

İşlem/Süreç
Yazılım bakım isteği oluştuğunda yapılması gereken işlemler: **a.** Değişiklik isteğine bir tanım numarası verilmesi, **b.** Değişiklik türünün sınıflandırılması, **c.** Değişiklik isteğinin kabul, red ya da daha ayrıntılı incelenmesi yönünde karar alınması, **d.** Değişiklik ile ilgili zaman/boyut/işgücü kestirimi yapılması, **e.** Değişiklik isteğinin önceliklendirilmesi **f.** Değişiklik isteğinin diğerleri ile birlikte zaman ve iş planına kaydedilmesi biçiminde özetlenebilir. Bu işlemlerin birçoğunda, değişiklik isteğini yapan kişi, kullanıcı temsilcileri, yazılım mühendisleri ve konu uzmanları birlikte çalışıp karar verirler.

2.1. Sorun Tanımlama Süreci

Denetim

Sorun tanımlama aşamasında, değişiklik isteğinin daha önceden yapılıp yapılmadığı denetlenmeli ve tek olduğu belirlenmelidir. Bu amaçla, daha önceki değişiklik istekleri taranır.

Çıktı

Bu sürecin temel çıktısı, doğrulanmış ve geçerlenmiş ve karar verilmiş Bakım İsteğidir. Bir veri tabanında saklanan bu isteğin ayrıntıları:

- Sorun ya da yeni gereksinimin tanımı,
- Sorun ya da gereksinimin değerlendirmesi,
- Başlangıç öncelik,
- Geçerleme verisi (Düzeltici bakım için),
- Başlangıç kaynak gereksinimi,
- Mevcut ve gelecekte kullanıcılar üzerindeki etkileri,
- Yararlı ve aksak yönler

2.1. Sorun Tanımlama Süreci

Ölçüt
Sorun tanımlama sırasında kullanılabilecek ölçütler: • Bakım isteklerinde kabul edilmeyen madde sayısı, • Gelen bakım istekleri sayısı ve • Sorun geçerleme için harcanan kaynak ve zaman biçimindedir.

2.2. Çözümleme Süreci

- Çözümleme süreci, veri tabanında saklanmış ve geçerlenmiş bakım isteğini girdi olarak alır, projeye ilişkin bilgi ve belgeleri kullanarak söz konusu isteğin yerine getirilmesi için gerekli genel planı yapar.

2.2. Çözümleme Süreci

Girdi
Çözümleme sürecinin girdileri: • Geçerlenmiş bakım isteği, • Başlangıç kaynak gereksinimleri ve diğer veriler ve • Mevcut proje yada sistem bilgi ve belgeleri biçimindedir

2.2. Çözümleme Süreci

İşlem/Süreç
Çözümleme süreci temel olarak iki aşamadan oluşur: Olurluk aşaması ve ayrıntılı çözümleme aşaması. Mevcut sisteme ya da projeye ilişkin yapısal belgelerin bulunmadığı durumlarda tersine mühendislik yöntemi kullanılır. Olurluk çalışması, • Değişikliğin etkisi, • Prototiplemeyi içeren seçenek çözümler, • Dönüştürme gereksinimlerinin çözümlemesi, • Güvenlik ve emniyet zorunlulukları • İnsan faktörleri, • Kısa ve uzun erimli maliyetler ve • Değişikliği yapmanın yararları bilgilerini içerir. Ayrıntılı çözümleme çalışmasında, değişiklik isteği için ayrıntılı gereksenim tanımlaması yapılır. Bu çalışmada etkilenen yazılım öğeleri (yazılım tanımları, yazılım gereksinimleri, tasarım, kod, vb) belirlenir. Yazılım öğelerinin değişmesi gereken kısımları belirlenir. En az üç düzeyli sınama stratejisi (birim sınaması, bütünleştirme sınaması ve kabul sınaması) tanımlanır. Bu aşamanın son çalışması ise, kullanıcıya en az etki yapacak şekilde değişiklik gereksinimlerinin nasıl karşılanacağı bilgilerini içeren "Başlangıç gerçekleştirim planı" dır.

2.2. Çözümleme Süreci

Denetim
Çözümleme çalışmasının denetiminde aşağıdaki işlemler yapılır. **a.** Gerekli proje yada sistem bilgi/belgelerine erişimin sağlanması (Ortam denetim organizasyonundan) **b.** Önerilen değişikliklerin ve çözümleme çalışmasının teknik ve ekonomik olurluğunun gözden geçirilmesi, **c.** Güvenlik ve emniyet konularının tanımlanması, **d.** Önerilen değişikliğin, mevcut yazılımla bütünleştirilmesinin dikkate alınması, **e.** Proje belgelerinin düzgün olarak günlendiğinin denetimi, **f.** Çözümleme belgelerinin düzgün olarak hazırlanmasının sağlanması, **g.** Sınama stratejilerinin uygun olarak belirlenmesi.

2.2. Çözümleme Süreci

Çıktı

Çözümleme çalışmasının çıktıları:

- Değişiklik isteklerine ilişkin olurluk çalışması,
- Ayrıntılı çözümleme raporu,
- İzlenebilirlik listesini içeren günlenmiş gereksinim tanımları,
- Başlangıç değişiklik listesi,
- Sınama stratejisi,
- Gerçekleştirim planı

2.2. Çözümleme Süreci

Ölçüt
Çözümleme çalışmasında kullanılabilecek ölçütler: • Gereksinimlerdeki değişiklik sayısı, • Belgeleme hata oranı, • Her işlev alanı için gerekecek işgücü ve • Toplam zaman biçimindedir.

2.3. Tasarım Süreci

- Tasarım aşamasında, değişiklikten etkilenebilecek tüm proje bilgi ve belgeleri üzerinde çalışma yapılıp söz konusu bilgi ve belgeler değişiklikle ilgili olarak günlenir.

2.3. Tasarım Süreci

Girdi

Tasarım çalışmasının girdileri:

- Çözümleme çalışması çıktıları
  1. Ayrıntılı çözümleme,
  2. Günlenmiş gereksinim tanımları,
  3. Başlangıç değişiklik listesi,
  4. Sınama stratejisi,
  5. Gerçekleştirim planı
- Sistem ve proje belgeleri ve
- Varolan kaynak kodları açıklamalar ve veritabanları

biçimindedir.

2.3. Tasarım Süreci

İşlem/Süreç
Tasarım için gerekli temel işlemler aşağıda belirtilmektedir. • Etkilenen yazılım modüllerinin tanımlanması, • Yazılım modül belgelerinin değiştirilmesi, • Yeni tasarım için, güvenlik ve emniyet konularını da içeren sınama senaryolarının hazırlanması, • İlişki sınamalarının tanımlanması, • Kullanıcı belgelerinin günleme gereksinimlerinin tanımlanması • Değişiklik listesinin günlenmesi

2.3. Tasarım Süreci

Denetim

Tasarım çalışmasında aşağıdaki denetim ortamı kurulmalıdır.
Tasarımın belirlenen standartlara uygunluğunun denetlenmesi,

- Güvenlik ve emniyet konularını içeren yeni tasarım belgesinin tasarım belgesinin ve bilgilerinin oluşturulmasının sağlanması,
- Sınama bilgilerinin günlenmesinin sağlanması,
- Gereksinimlerden tasarıma izlenebilirliğin sağlanası.

2.3. Tasarım Süreci

Çıktı

Bakım tasarımı çalışmasının çıktıları:

- Gözden geçirilmiş değişiklik listesi,
- Günlenmiş tasarım
- Günlenmiş sınama planları,
- Günlenmiş ayrıntılı çözümleme,
- Günlenmiş gereksinimler,
- Gözden geçirilmiş gerçekleştirim planı
- Risk ve kısıtlar listesi

biçimindedir.

2.3. Tasarım Süreci

Ölçüt
Tasarım çalışması için kullanılabilecek ölçütler aşağıda verilmektedir. • Yazılım karmaşıklığı, • Tasarım değişiklikleri • Her işlev alanı için gerekecek işgücü, • Toplam zaman, • Sınama yönerge ve plan değişiklikleri, • Önceliklendirmedeki hata oranları, • Varolan kodda, eklenen, çıkarılan ve değiştirilen satır sayısı, • Uygulama sayısı.

2.4. Gerçekleştirim Süreci

- Gerçekleştirim süreci, temel olarak tasarım çıktılarını ve kaynak kodları girdi olarak almakta ve değişiklik isteğini gerçekleştiren kod parçaları ile günlenmiş yazılım kodlarını üretmektedir.
- Günlenmiş yazılıma ilişkin sınama bilgi ve belgelerinin ve eğitim belgelerinin üretimi de bu süreçte yapılmaktadır.

2.4. Gerçekleştirim Süreci

2.4. Gerçekleştirim Süreci

Girdi
Gerçekleştirim sürecinin girdileri: • Tasarım çalışması sonuçları, • Varolan kaynak kodlar, açıklamalar, belgeler ve • Proje ve sistem belgeleri, biçimindedir.

2.4. Gerçekleştirim Süreci

İşlem/Süreç
Gerçekleştirim sürecinin dört ana işlemi vardır: • Kodlama ve birim sınama • Bütünleştirme" Risk çözümleme • Sınama hazırlığı gözden geçirme • Kodlama işleminde, değişiklik isteğini karşılayan yazılım kodları, varolan yazılıma eklenmektedir. İşlem sonucunda elde edilen yeni, değişmiş modüllere birim sınama uygulanmaktadır. Birim sınama işlemini, bütünleştirme sınama izlemekte, tüm sistem yeniden sınanmaktadır.Uygulamadaki riskleri gidermek amacıyla, gerçekleştirim aşamasında sürekli risk çözümleme yapılmaktadır.

2.4. Gerçekleştirim Süreci

Denetim

Gerçekleştirim sürecinde oluşturulacak denetim yapısı, aşağıdaki özellikleri sağlamalıdır:

- Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması,
- Birim ve bütünleştirme sınamaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması,
- Sınama belgelerinin günlenmesi ve oluşturulmasının sağlanması,
- Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin sağlanması,
- Yeni yazılımın, yazılım ortam yönetimi altında kaydedilmesi ve denetlenmesinin sağlanması,
- Teknik ve eğitim belgelerinin günlenmesinin sağlanması,
- Tasarımdan koda izlenebilirliğin sağlanması

2.4. Gerçekleştirim Süreci

Çıktı

Gerçekleştirim süreci aşağıdaki çıktıları vermelidir:

- Günlenmiş Yazılım,
- Günlenmiş tasarım bilgi/belgeleri,
- Günlenmiş sınama belgeleri,
- Günlenmiş kullanıcı belgeleri,
- Günlenmiş eğitim kılavuzları,
- Riskler ve kullanıcılara etkileri,
- Sınama hazırlığı gözden geçirme rapor.

Ölçüt

Gerçekleştirim çalışmasında kullanılabilecek ölçütler:
" Değişiklik oranı ve
" Hata oranı
biçimindedir.

2.5. Sistem Sınama Süreci

- Değişikliklerin varolan yazılıma yansıtılmasından sonra elde edilen yeni yazılım sürümünün belirlenen standartlara uygun olarak tümüyle bütünleşik sistem üzerinde sınamaların yapılması gerekmektedir.
- Sistem sınamalarının, kullanıcı ve üretici ekiplerin tanıklığında bağımsız bir yapı tarafından gerçekleştirilmeleri önerilmektedir.

2.5. Sistem Sınama Süreci

2.5. Sistem Sınama Süreci

Girdi
Sistem sınama sürecinin girdileri: **a.** Sınama hazırlık raporu **b.** Belgeler • Sistem sınama planları, • Sistem sınamaları, • Sistem sınama yönergeleri, • Kullanıcı kılavuzları, • Tasarım **c.** Günlenmiş sistem biçimindedir.

2.5. Sistem Sınama Süreci

İşlem/Süreç
Sistem sınama, tümüyle bütünleşik bir sistem üzerinde yapılmalıdır. Bu aşamada, işlevsel sistem sınama, arayüz sınama, regresyon sınama ve sınama hazırlık raporunun gözden geçirilmesi işlemleri yapılır. Denetim: Sistem sınamaları, üretici ve kullanıcılardan bağımsız bir grup tarafından gerçekleştirilmelidir. Yazılım kodları ve her türlü bilgi belge, yazılım ortam yönetimi tarafından saklanır.

2.5. Sistem Sınama Süreci

Denetim
Gerçekleştirim sürecinde oluşturulacak denetim yapısı, aşağıdaki özellikleri sağlamalıdır: • Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması, • Birim ve bütünleştirme sınamaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması, • Sınama belgelerinin günlenmesi ve oluşturulmasının sağlanması, • Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin sağlanması, • Yeni yazılımın, yazılım ortam yönetimi altında kaydedilmesi ve denetlenmesinin sağlanması, • Teknik ve eğitim belgelerinin günlenmesinin sağlanması, • Tasarımdan koda izlenebilirliğin sağlanması

2.5. Sistem Sınama Süreci

Çıktı

Gerçekleştirim süreci aşağıdaki çıktıları vermelidir:

- Günlenmiş Yazılım,
- Günlenmiş tasarım bilgi/belgeleri,
- Günlenmiş sınama belgeleri,
- Günlenmiş kullanıcı belgeleri,
- Günlenmiş eğitim kılavuzları,
- Riskler ve kullanıcılara etkileri,
- Sınama hazırlığı gözden geçirme rapor.

Ölçüt

Gerçekleştirim çalışmasında kullanılabilecek ölçütler:

- Değişiklik oranı ve
- Hata oranı

biçimindedir.

2.6. Kabul Sınaması Süreci

- Kabul sınaması süreci, kullanıcılar ya da kullanıcı temsilcileri tarafından gerçekleştirilen bir süreçtir. Kullanıcıların, değişiklikleri içeren yeni yazılımı sınamaları ve kabul etmeleri beklenmektedir.

2.6. Kabul Sınaması Süreci

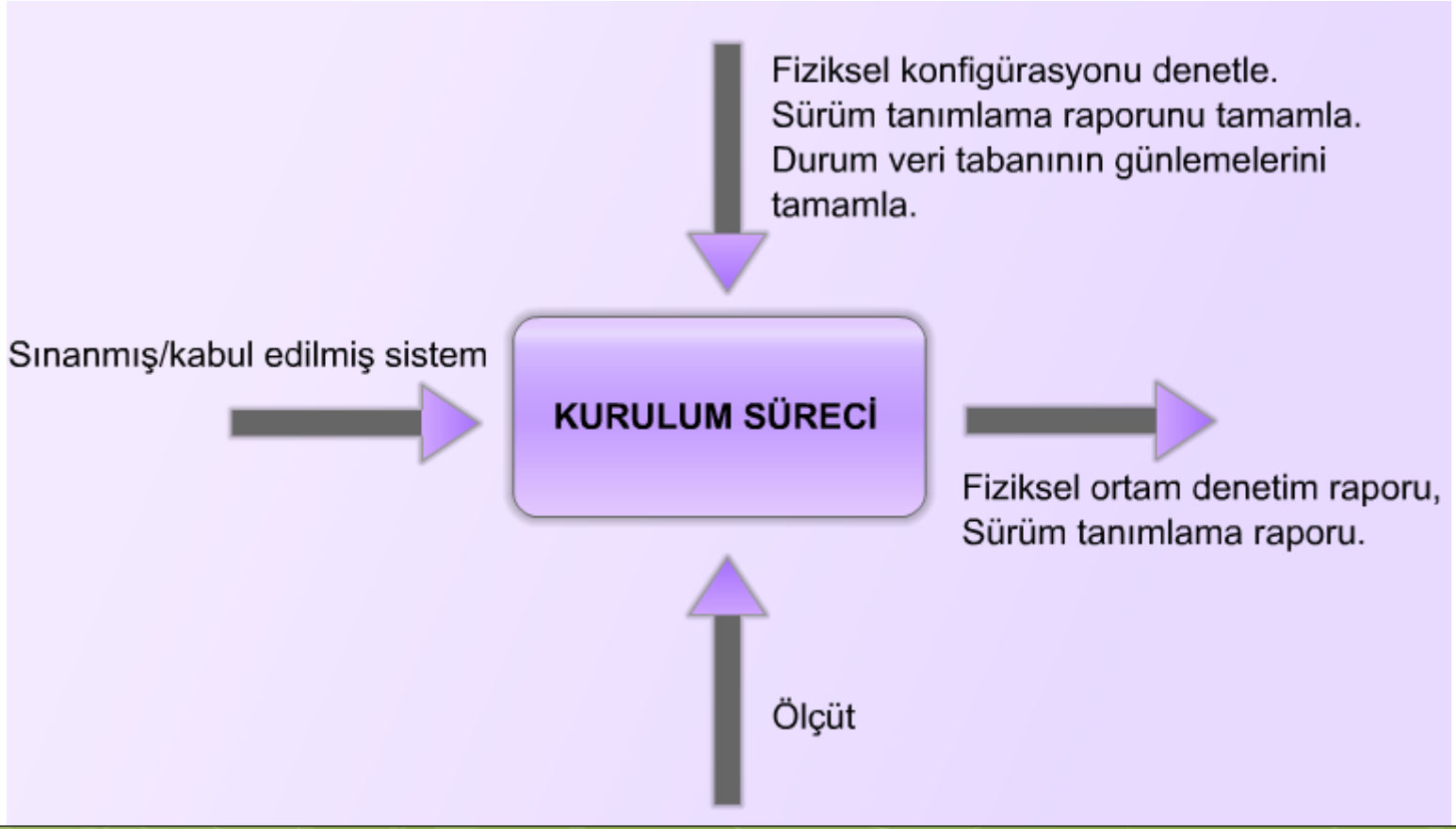
Girdi
Kabul sınama sürecinin girdileri: **a.** Gözden geçirilmiş sınama hazırlık raporu, **b.** Tümüyle bütünleşik sistem, **c.** Kabul sınama planları, **d.** Kabul sınamaları ve **e.** Kabul sınama yönergeleri biçimindedir.
İşlem/Süreç
Kabul sınaması işlemleri: **a.** İşlevsel kabul sınamalarının yapılması, **b.** Birlikte çalışabilirlik sınaması, **c.** Regresyon sınaması biçimindedir.

2.6. Kabul Sınaması Süreci

Denetim
Kabul sınamaları sırasında denetimi aşağıdaki işlemleri içermektedir. a. Kabul sınamalarının uygulanması, b. Sınama sonuçlarının raporlanmasının sağlanması, c. İşlevsel denetim yapılması, d. Yeni sistemin oluşturulması, e. Kabul sınama belgelernin yazılım konfigürasyonuna yerleştirilmesi.
Çıktı
Kabul sınamalarının çıktıları, yeni sistem, işlevsel konfigürasyon denetim raporu ve kabul sınaması raporudur.
Ölçüt
Bu aşamada kullanılabilecek ölçütler: üretilen ve düzeltilen hata oranlarıdır.

2.7. Kurulum Süreci

- Kurulum süreci, geliştirilen ya da değiştirilmiş yeni yazılım sürümünün, uygulama sahasına aktarılma işlemlerini içerir.



2.7. Kurulum Süreci

Girdi
Bu sürecin temel girdisi, tümüyle sınanmış ve kabul edilmiş yeni yazılım sürümüdür.
İşlem/Süreç
Bu sürecin işlemleri: a. Fiziksel ortam denetiminin yapılması, b. Kullanıcıların bilgilendirilmesi, c. Varolan sistemin yedeklerinin alınması, d. Kullanıcı tarafında kurulum ve eğitimlerin yapılması biçimindedir.

2.7. Kurulum Süreci

Denetim
Denetim işlemleri: a. Fiziksel ortam denetiminin yapılması, b. Sistem ile ilgili bilgi ve belgelerin kullanıcıya ulaştırılması, c. Sürüm Tanımlama raporunun tamamlanması ve d. Yazılım konfigürasyon ortamına aktarımın sağlanması alanlarını içermektedir.
Çıktı
Bu sürecin temel çıktıları, fiziksel ortam denetim raporu ve Sürüm tanımlama raporudur.
Ölçüt
Bu süreçte kullanılabilecek ölçüt, belgeleme değişiklikleridir.

Alınan Dersler 1

- Kurulum planlamasında mevsimsel koşulların dikkate alınması önemlidir.
- Örneğin, kış mevsiminde yoğun kar alan doğu bölgelerindeki sahalara kurulum zaman zaman olanaksızdır.
- Bu nedenle, kurulum planlamasında, bu tür bölgeler bahar ya da yaz ayları içerisinde planlanmalıdır.

Alınan Dersler 2

- Kurulum, yalnızca uygulama yazılımı sürümlerinin yüklenmesini içermemekte, zaman zaman teknolojik alt yapı değişebilmekte ve yeni sistem yazılımı sürümlerinin yüklenmesi gerekebilmektedir.
- Bu nedenle, kurulum elemanlarının, gerekli teknik bilgilerle donatılması ve teknolojik değişimleri izlemelerinin sağlanması gerekmektedir.

Alınan Dersler 3

- Uç kullanıcılar zaman zaman kendi bilgisayarlarına dışarıdan getirdikleri ya da internet ortamından sağladıkları yazılımları yüklerler.
- Virüs riski taşıyan bu tür yazılımlara karşı kullanıcıları sürekli uyarmak ve anti-virüs yazılımlarının kullanımını özendirmek gerekmektedir.

Alınan Dersler 4

- Ülkemizde kullanılan yazılım üretim yöntemleri düşünüldüğünde, bilinen anlamda bakım yapılmadığı, yazılımları hazırlayan kişilerin yıllarca kendi yazdıkları yazılıma yamalar yaptıkları, yazılımı usta-çırak ilişkisi içerisinde başkalarına devretmeye çalıştıkları ve çalışan yazılıma ilişkin elde düzgün teknik belgelerin olmadığı bir ortamla karşılaşılmaktadır.

Alınan Dersler 4-5

- Bu durum zaten pahalı olan bilişim iş gücünü verimsiz kullanmamıza neden olmaktadır.
- Kullanıcı ve yerinde destek elemanları arasındaki tüm iletişimin kayıt altına alınmasının sağlanması ve izlenmesi gerekmektedir.

Sorular

1. Kurulum işlemlerini belirtiniz.

2. Yerinde destek ekibinin görev ve sorumluluklarını tanımlayınız.

3. Uygulamakta olduğunuz bir proje için yaptığınız ya da yapmayı düşündüğünüz kurulum işlemlerini açıklayınız.

4. Geçiş dönemini tanımlayınız.

5. Bir uygulama yazılımının ömrü nasıl belirlenir?

6. Yazılım Bakımını tanımlayınız, bakım türlerini açıklayınız.

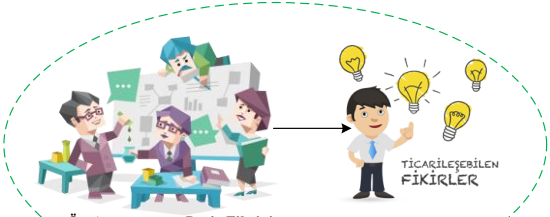
7. Bakım süreçlerinin her birinde oluşabilecek risklere birer örnek veriniz.

Genel

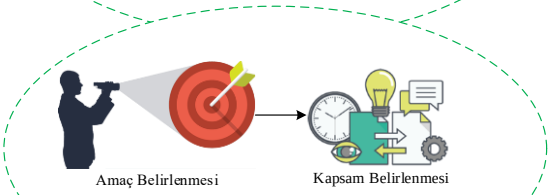
- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach



Ön Araştırma ve Proje Fikrinin Belirlenmesi



Proje Fikrinin Analizi

- Projenin Plan Kapsamı
- Proje Zaman-İş Planı
- Proje Ekip Yapısı
- Önerilen Sistemin Teknik Tanımları
- Kullanılan Özel Geliştirme Araçları ve Ortamları

- Proje Standartları, Yöntem ve Metodolojiler
- Kalite Sağlama Planı
- Konfigürasyon Yönetim Planı
- Kaynak Yönetim Planı
- Eğitim Planı
- Test Planı
- Bakım Planı

Proje Plan ve Aşama Analizi

- Mevcut Sistem İncelemesi
- Gereksenen Sistemin Mantıksal Modeli
- Arayüz (Modül) Gerekleri
- Belgeleme Gerekleri

Sistem Çözümleme ve İnceleme Aşaması

- Genel Tasarım Bilgileri
- Veri Tasarımı
- Süreç Tasarımı
- Ortak Alt Sistemlerin Tasarımı

Sistem Tasarım Aşaması

- Yazılım Geliştirme Ortamları
- Kodlama Stili
- Program Karmaşıklığı
- Olağan Dışı Durum Çözümleme
- Kod Gözden Geçirme

Sistem Gerçekleştirim Aşaması

- Sınama Kavramları
- Doğrulama ve Geçerleme Yaşam Döngüsü
- Sınama Yöntemleri
- .Sınama ve Bütünleştirme Stratejileri

- Sınama Planlaması
- Sınama Belirtimleri
- Yaşam Döngüsü Boyunca Sınama Etkinlikleri

Doğrulama ve Geçerleme Aşaması

- Kurulum
- Yerinde Destek Organizasyonu
- Yazılım Bakımı

Bakım ve Destek Aşaması

SONUÇ **KAYNAKLAR**

Gereksinim Verisi Toplama

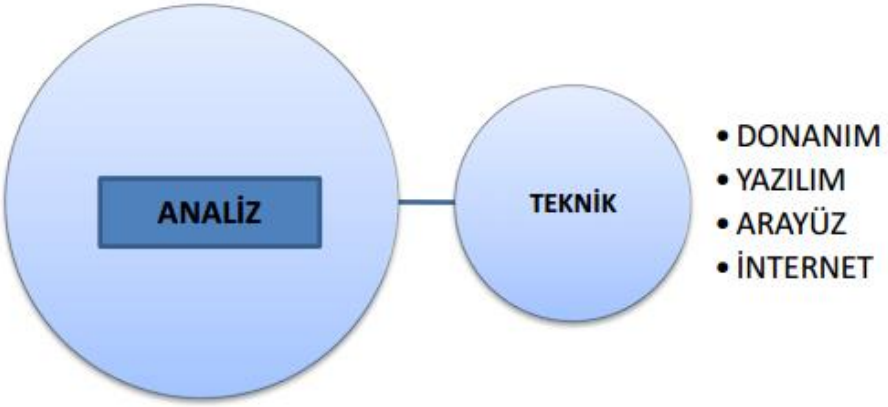
Yazılım mühendisliği projelerinin geliştirilmesi sürecinde en önemli unsurlardan birisi de mevcut sistemin incelenmesidir. Mevcut sistemlerin incelenmesi bizlere gereksinim analizinin doğru bir biçimde yapılabilmesi için somut spesifikasyonlar sağlamaktadır. Genel olarak müşteri ile görüşmeler ve mevcut sistemin incelenmesi sonucunda gereksinim analizi dökümanı üretilir. Ayrıca önemli bir diğer unsur ise yeni sistem için mantıksal bir modelin ortaya konulmasıdır.

Mevcut sistemin incelenmesi sırasında kullanılabilecek temel yöntemler aşağıdaki gibi özetlenebilir;

1. Sorma
2. Psikolojik Türetme Teknikleri
3. İstatistiksel Teknikler

1. Sorma Yöntemi Örneği

Örnek olarak verilecek sorma yöntemi bir sistem hakkındaki teknik gereksinimlerdir. Teknik gereksinimler; teknik kısıtları veya koşulları sistemin çalışması prensibinde tanımlar. Bu gereksinimler aynı zamanda servis kalitesi gereksinimleri olarak da bilinirler.



SORULAR:

Donanım

- Mevcut donanım potansiyeli yeni uygulamayı karşılayabilecek mi?
- Mevcut donanım dışında yeni donanım satın alınması gerekiyor mu?
- Mevcut yazılım ortamında kullanıcıların işletim sistemleri nedir?(Windows xp..)
- Serverın güvenliği için gerekli olan lisanslar mevcut mu?
- Hangi virüs programı sisteminiz de kurulu?
- Sistemin performansını etkileyecek donanımsal sorunlarınız var mı?

Yazılım

- Mevcut uygulama dışında başka hangi uygulamalar kullanımda?
- Daha önce kullanmış olduğunuz uygulama var mı?
- Mevcut uygulama daha büyük bir sistemin bileşeni midir?
- Alt sistemlerin birbirleriyle etkileşimi nasıl?

- Uygulama web tabanlı mı mobil mi masaüstü uygulaması olarak mı geliştirilecek?

Arayüz

- Ekran tasarımının nasıl olmasını istersiniz? (Rengi.. nasıl olmalı)
- Uygulamada kullanılmasını istediğiniz logonuz var mı?
- Admin panelinin dinamik bir şekilde yapılandırılmasını istiyor musunuz?

İnternet

- Sistemin yükünü karşılayabilecek ağ yapısı mevcut mu?
- Sistem İnternet üzerinden güncellenebilsin mi?
- Sistem farklı ağ yapılarına sahip mi?
- Ağ da oluşabilecek sorun hızlı bir şekilde çözülebiliyor mu?

Bu doküman örnek olması açısından verilmiş bir belgedir. Geliştirilecek projeye göre bu sorma listesi sorularının farklı olabileceği açıktır. Mevcut sistemin analizi için İstatistiksel Teknikler ve Psikolojik Türetme Teknikleri de kullanılmaktadır.

Kaynaklar: Bilge Adam Bilişim Hizmetleri Yazılım Kalitesi ve Süreç Yönetimi Departmanı - İş Analizi Gereksinim Soru Listesi Dökümanı

Nesne Diyagramları

Nesne diyagramları sınıf diyagramlarından türetilir, böylece nesne diyagramları sınıf diyagramlarına bağımlıdır. Nesne diyagramları, bir sınıf diyagramının bir örneğini temsil eder. Temel kavramlar sınıf diyagramları ve nesne diyagramları için benzerdir. Nesne diyagramları aynı zamanda bir sistemin statik görünümünü de temsil eder, ancak bu statik görünüm belirli bir anda sistemin anlık görüntüsüdür. Nesne diyagramları, bir nesne kümesini ve ilişkilerini bir örnek olarak işlemek için kullanılır.

Niçin Kullanılır?

Bir diyagramın amacı pratik olarak uygulanması için açıkça anlaşılmalıdır. Nesne diyagramlarının amacı sınıf diyagramlarına benzer. Aradaki fark sınıf diyagramının sınıflardan ve onların ilişkilerinden oluşan soyut bir modeli temsil etmesidir. Ancak bir nesne diyagramı, belirli bir andaki somut bir örneği temsil eder. Bu, nesne diyagramının gerçek sistem davranışına daha yakın olduğu anlamına gelir. Bu açıdan bakılınca kullanılmasındaki amaç, belirli bir anda bir sistemin statik görünümünü elde etmektir.

- İleri ve geri mühendislik.
- Bir sistemin nesne ilişkileri
- Bir etkileşimin statik görünümü.
- Nesne davranışını ve bunların ilişkilerini pratik perspektiften anlama

Nesne Diyagramları Nasıl Çizilir?

Bir nesne diyagramının bir sınıf şemasının bir örneği olduğunu zaten dersimizde de tartışmıştık. Bu, bir nesne diyagramının bir sınıf şemasında kullanılan örneklerinden oluştuğunu ifade eder. Dolayısıyla her iki diyagram da aynı temel öğelerden oluşur ancak farklı biçimdedir. Sınıf diyagramında öğeler mavi baskıyı temsil etmek için soyut formda ve nesneler diyagramında öğeler gerçek dünya nesnesini temsil etmek için somut biçimde bulunmaktadır. Belirli bir sistemi yakalamak için sınıf diyagramlarının sayısı sınırlıdır. Ancak nesne diyagramlarını düşünürsek doğada benzersiz olan sınırsız sayıda örnek elde edebiliriz. Dolayısıyla yalnızca sistem üzerinde etki yaratan örnekler düşünüldü. Yukarıdaki tartışmadan açıkça görüleceği gibi, tek bir nesne şeması gerekli tüm örnekleri yakalayamaz veya bir sistemin tüm nesnelerini belirtemez. Bu açıdan bakılınca nesne diyagramları çizilirken dikkat edilecek durumlar şunlardır:

- İlk olarak, sistem analiz edilmelidir ve hangi örneklerin önemli veri ve ilişkiye sahip olduğuna karar verilmelidir.
- İkinci olarak, yalnızca işlevselliği kapsayan örnekler göz önünde bulundurulmalıdır.
- Üçüncü olarak, örnek sayısı sınırsız olduğundan bazı optimizasyon işlemleri yapılmalıdır.

Bir nesne diyagramları çizmeden önce aşağıdaki şeyler hatırlanmalı ve anlaşılmalıdır:

- Nesne diyagramları nesnelerden oluşur.
- Nesne diyagramındaki bağlantı, nesneleri birbirine bağlamak için kullanılır.
- Nesneler ve bağlantılar, bir nesne diyagramı oluşturmak için kullanılan iki elementtir.

Bundan sonra diyagramın yapımına başlamadan önce aşağıdaki şeyler kararlaştırılacaktır:

- Nesne diyagramının amacı belirtilmek için anlamlı bir adı olmalıdır.
- En önemli unsurlar belirlenmelidir.
- Nesneler arasındaki ilişki netleştirilmelidir.
- Nesne şemasına dahil etmek için farklı öğelerin değerleri yakalanmalıdır.
- Netliğin gerekli olduğu noktalarda doğru notlar eklenmelidir.

Nesne Diyagramları Nerede Kullanılır?

- Bir sistemin prototipini yapmada.
- Tersine mühendislik uygulamalarında.
- Karmaşık veri yapılarını modellemede.
- Sistemi pratik perspektiften anlamada.

Örnek Senaryo

1) Hasta, randevu öncesi, hastaneye gelir.

2) Hasta, randevusunu, görevliye bildirir.

3) Görevli, muayene ücretini talep eder.

4) Hasta, muayene ücretini, görevliye öder.

5) Görevli, ödeme makbuzunu, hastaya verir.

6) Görevli, hastayı doktora yönlendirir.

7) Hasta, doktorun odasına gider.

8) Hasta, kapıdaki ekranda ismini görürse, odaya girer. Diğer durumda, isminin ekranda görünmesini bekler.

9) Doktor, sıradaki hastayı çağırır.

10) Doktor, hastaya, teşhis amaçlı bazı sorular sorar.

11) Doktor, hastayı muayene eder.

12) Doktor gerekli görürse, farklı laboratuvarlardan tetkik ister. 13) Laboratuvar görevlisi, sıradaki tetkik isteğini alır.

14) Laboratuvar görevlisi, hastayı çağırır ve numune alır.

15) Laboratuvar görevlisi, tetkik üzerinde çalışır ve raporu yayınlar.

16) Sistem, hastayı ve doktorunu e-mail ile uyarır. E-mail içinde rapor linki vardır. Hasta raporunu, web üzerinden inceleyebilir.

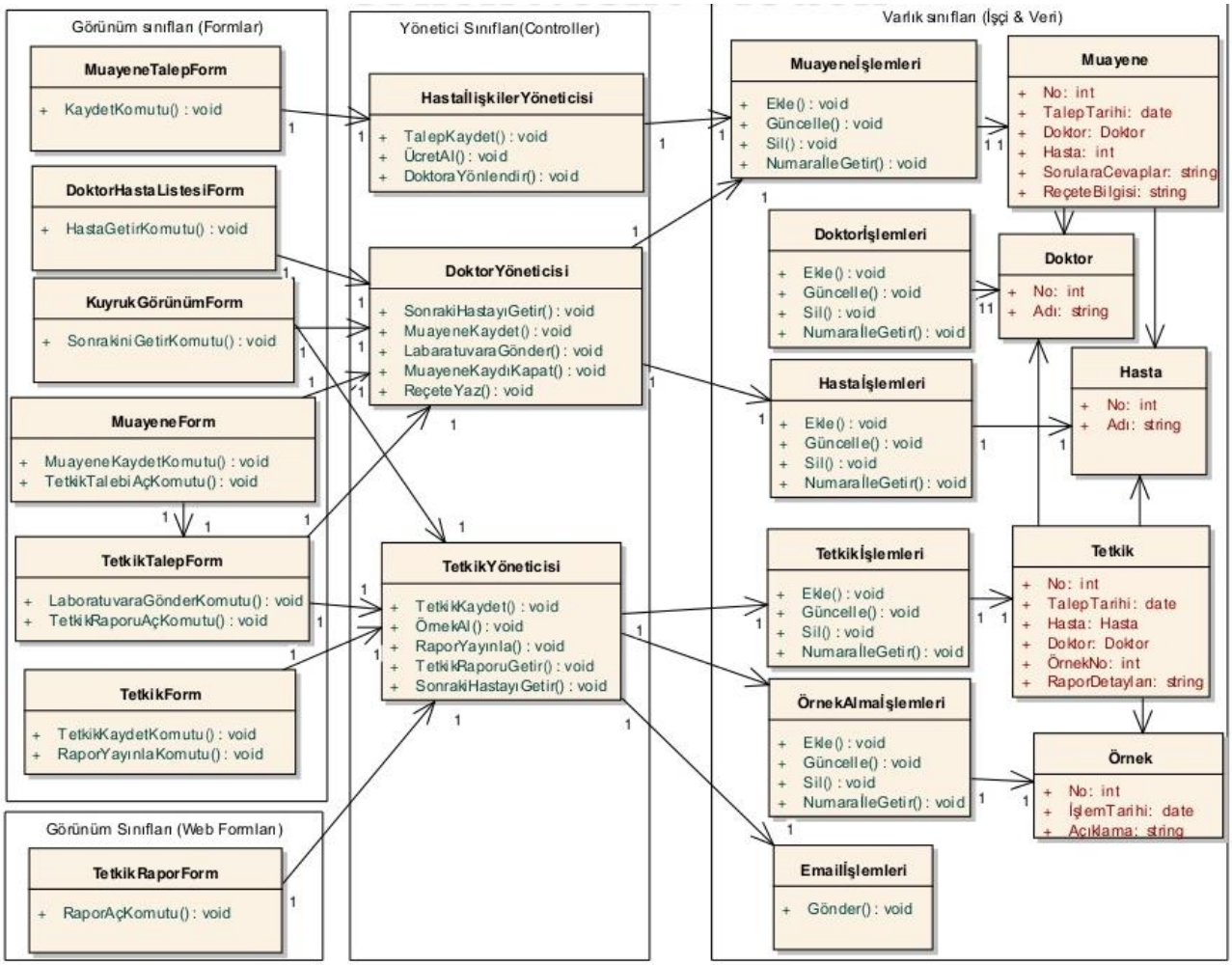
17) Doktor tetkikleri inceler.

18) Doktor ilaç yazar.

19) Doktor muayene kaydını kapatır.

Diğer senaryolar

- Randevusuz gelen hasta
- Sadece kontrol için gelen hasta
- Acil hasta (Ambulans)
- Ameliyat gerektiren hasta



Kaynaklar

https://www.tutorialspoint.com/uml/uml_object_diagram.htm

https://www.lucidchart.com/pages/uml/object-diagram

https://www.slideshare.net/gpbilisim/yazlm-gereksinim-muhendisligi

https://sites.google.com/site/modernyazilimgelistirme/uml/uml-diyagramlari/paket-ve-nesne-diyagramlari

YMH 114 YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

Konular

- Ders içeriği, başarı değerlendirme
- YM kavramları, süreçler, modeller, yöntemler
- Yazılım Gereksinim Çözümlemesi
- Yazılım Tasarımı
- Yazılım Gerçekleştirimi
- Test ve Bakım
- Yazılım Geliştirme Yöntem Bilimleri
- Risk, Kalite ve Proje Yönetimi
- UML

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

Araştırma ve Proje

- Araştırma ve Sunum

(Gereksinim, Tasarım, Programlama (Kodlama/Hata Ayıklayıcı) (Kullanıcı Önyüz), Test, ProjeYönetimi, Düzenleşim (Konfigürasyon)/Değişim Yönetimi konularında kullanılan araç ve gereçleri araştır)

- Proje

(Bir konu üzerinde yazılım mühendisliği aşamalarının UML kullanılarak uygulanması)

Kullanılacak Yazılımlar

- Proje raporlama ve dökümantasyon işlemlerinde ihtiyaç duyulabilecek bazı araçlar şöyledir;
- MS Office Word, (proje raporu yazımı)
- MS Office Excel (Tablolama vb.)
- Adobe Photoshop (şekil çizimlerinde vb.)
- IBM RSA (Retional Software Architect) (UML diyagram çizimlerinde)
- MS Visio (Akış şeması çizimlerinde vb.)

Yazılım Nedir

Yazılım

– Tanımlanmış bir işlevi yerine getiren,
– Girdi ve Çıktıları olan,
– Herhangi bir donanım üzerinde çalışan,
– Bilgisayar programı veya programlarından ve
– Kullanım ve bakım kılavuzları gibi belgelerden oluşan

bir üründür.

Yazılım Mühendisliği (YM) - Nedir

"Mühendislik eylemlerinin, (Geliştirme, İşletme, ve Bakım), disiplinli, sistematik ve nicelikli bir şekilde yazılıma uygulanması"

YM – Önemi

- Yazılımın hayatımıza girmediği yer var mı?
- Yazılımsız hayat nasıl olurdu?
- Yazılım ve Eğitim
- Yazılım ve Ekonomi
- Yazılım ve Haberleşme
- Yazılımın verimliliğe katkısı
- Yazılımın kültüre etkisi

YM – Tarihçesi

- İlk Bilgisayarlar ve Makine ve Assembly Dili
- İşletim Sistemleri ve Anabilgisayarlar (Mainframe),
- Kart okuyucuları
- DOS ve PC'ler
- Derleyici (Compiler) ve Yorumlayıcılar (Interpreter) ve Yeni nesil yazılım dilleri
- Windows
- Yarı İletken teknolojisinin fiyat ve boyutlara etkisi
- Veri haberleşmesindeki gelişmeler
- **Internet**

Yazılım Geliştirme İstatistikleri

- Tipik yazılım projesinin geliştirilmesi 1-2 yıl sürüyor ve en azından 500.000 kod satır içeriyor
- Tüm projelerin yalnız %70-80'i başarıyla tamamlanıyor
- Tüm geliştirme sürecinde her birey günde ortalama 10 satırdan az kod yazıyor
- Geliştirme süresince her 1000 kaynak kod satırında 50-60 hata bulunuyor (satışa sunulmuş sistemde hata sayısı 4/1000'e düşüyor)

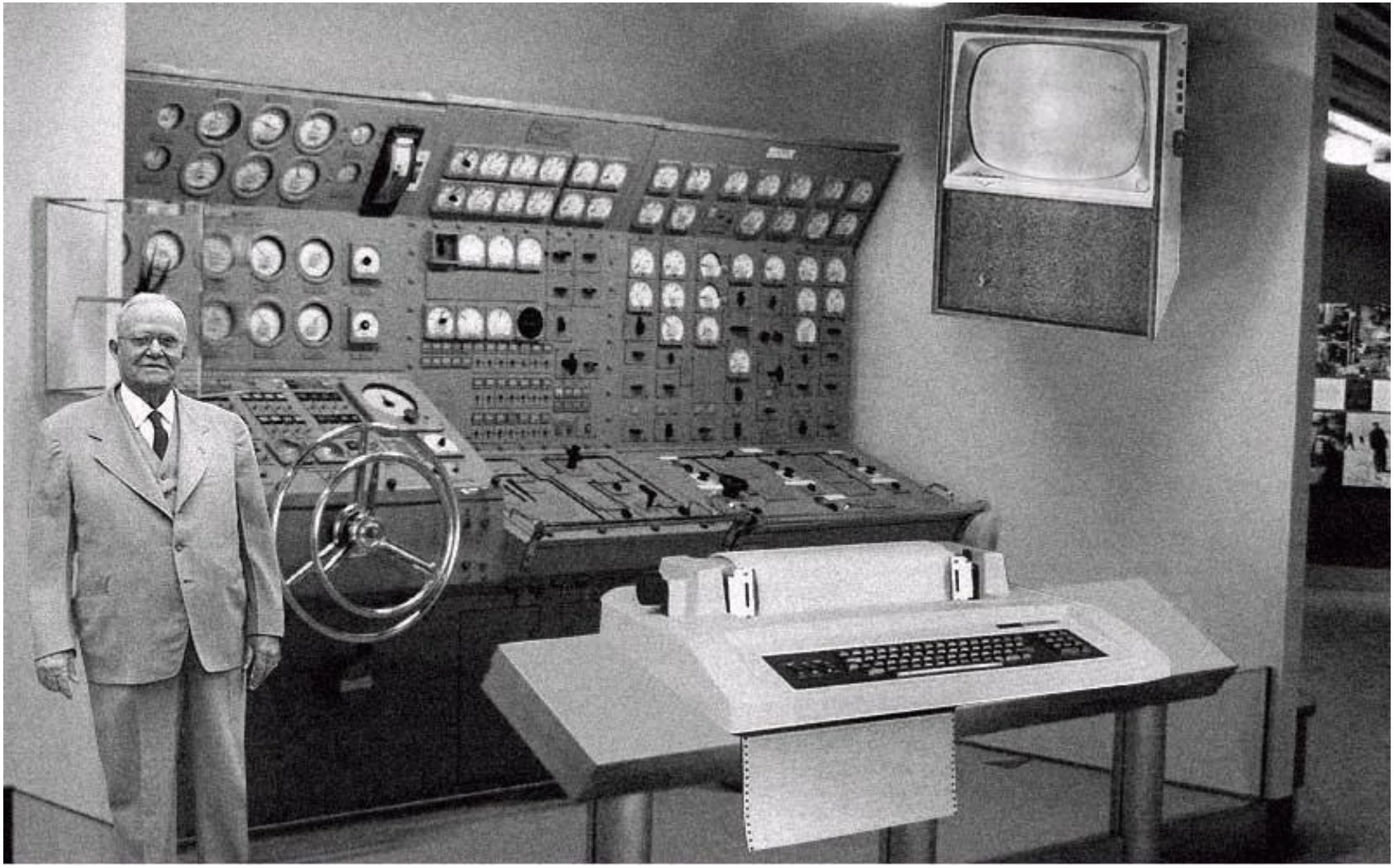
Yazılım Problemleri (Krizleri)

- Tasarlanan zamanın gerisinde kalma
- Bütçeyi aşma
- Düşük Kalite
  - Güvenilir olmayan yazılım
  - Kullanıcı taleplerinin karşılanmasında yetersizlik
  - Sürekliliğinin sağlanmasındaki zorluk

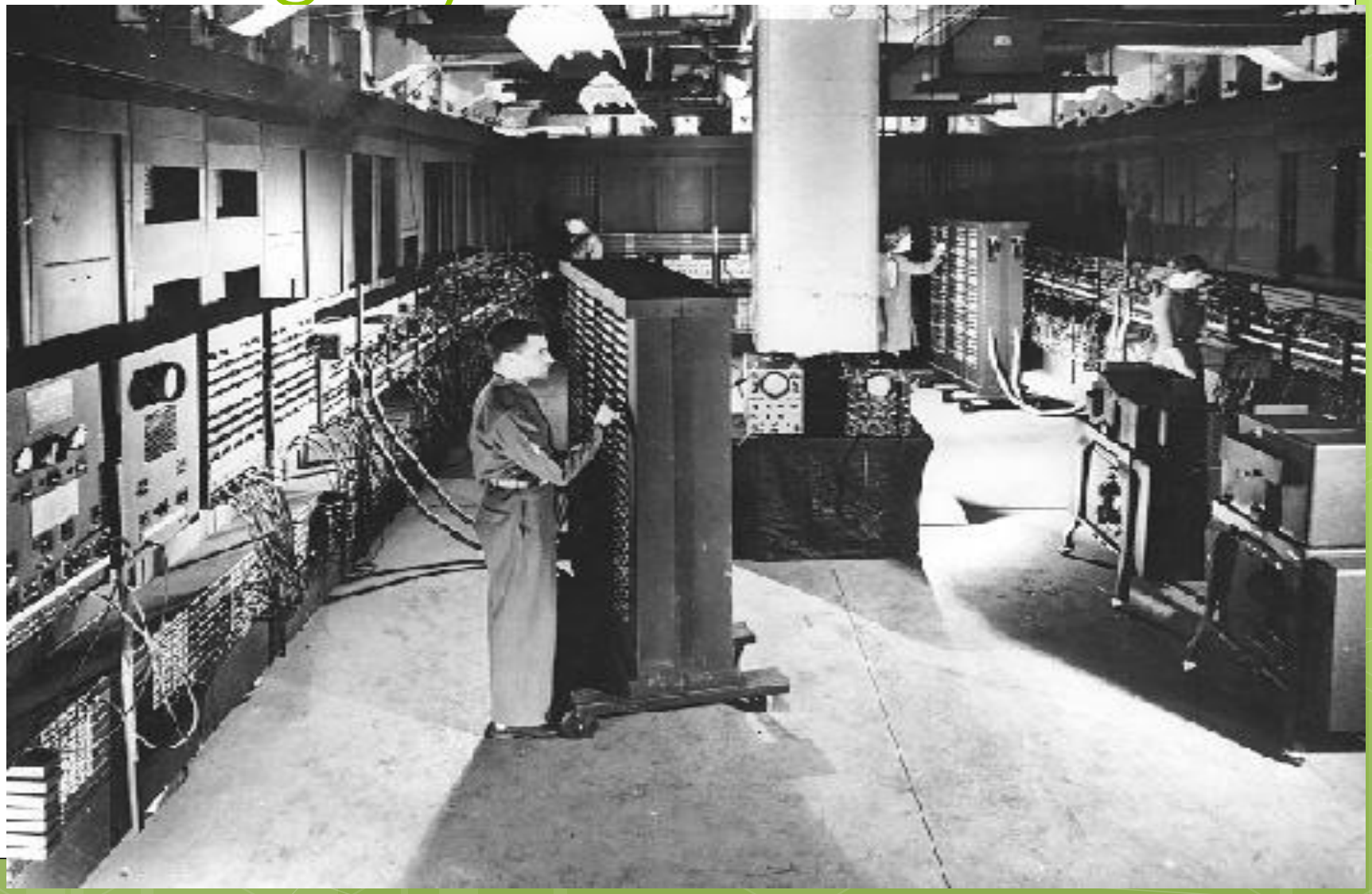
Yazılım-Donanım Evrimi

- Erken Yıllar 1950-1960 ilk bilgisayarlar, makine dili, 3-4 şirket
- Yazılımlar; kullanıcı ile birebir iletişimde bulunmayan, işlerin toplu olarak verilip, yalnızca yazıcı çıktılarının alındığı biçimde geliştirilmekte idi.
- Ayrıca, yazılımlar bu günkü gibi ürün tarzında değil, kuruluşa özel olarak geliştirilmekte idi.

İlk Bilgisayarlar



İlk Bilgisayarlar



Yazılım-Donanım Evrimi

- İkinci Dönem 1960-1975 Ana çatı, çok kullanıcılı sistemler,
  Veri tabanı yönetimi
- Çok kullanıcılı, gerçek zamanlı yazılımlar üretilmeye başlandı.
- VTYS önce yapılandırılmış dosya sistemleri ile ortaya çıktı.
- Yavaş yavaş ürün türü yazılımlar ortaya çıkmaya başladı.

Yazılım-Donanım Evrimi

- Üçüncü Dönem 1975-1990 Süper bilgisayarlar, Kişisel bilgisayarlar
- Açık sistem mimarisinin tanıtıldığı bu dönemde ürün bazlı yazılımlar oldukça yaygınlaştı.
- Kişisel bilgisayarlar yaygınlaşarak evlere girmeye başladı.
- Ağ yapısının gelişmesi ve güçlenmesiyle, dağıtılmış yazılım sistemleri geliştirilmeye başlandı.
- Yapay zeka teknolojisinin gelişmesiyle "akıllı uygulama yazılımları" üretilmeye başlandı

Yazılım-Donanım Evrimi

- Dördüncü Dönem 1990 - Yapay Zeka, Gömülü Sistemler, Paralel Sistemler, Yazılım Kaliteleri
- Uzman sistem yazılımları oldukça gelişmiş ve mikro-bilgisayarlar üzerinde yaygınlaşmıştır.
- "Yazılımda Kalite" olgusu önem kazanmış ve yazılım ile ilgili standartlar olgunlaşmaya başlamıştır.
- Yazılım üretimi ve ürünlerinin değerlendirilmesi amacıyla kurumlar oluşmaya başlamıştır.

Programlama Dillerinin Seviyeleri

1.Kuşak **Makine Dili**

10101110 10010001

2. Kuşak **Assembler**

8085, Z80, 68000, vs

3.Kuşak **Üst Seviye Diller**

Pascal Coral66 Basic

Bildirimsel

LISP Hope Prolog

Nesneye Yönelik Diller

Smalltalk, C++, Java

4. Kuşak **Veri Yapısal**

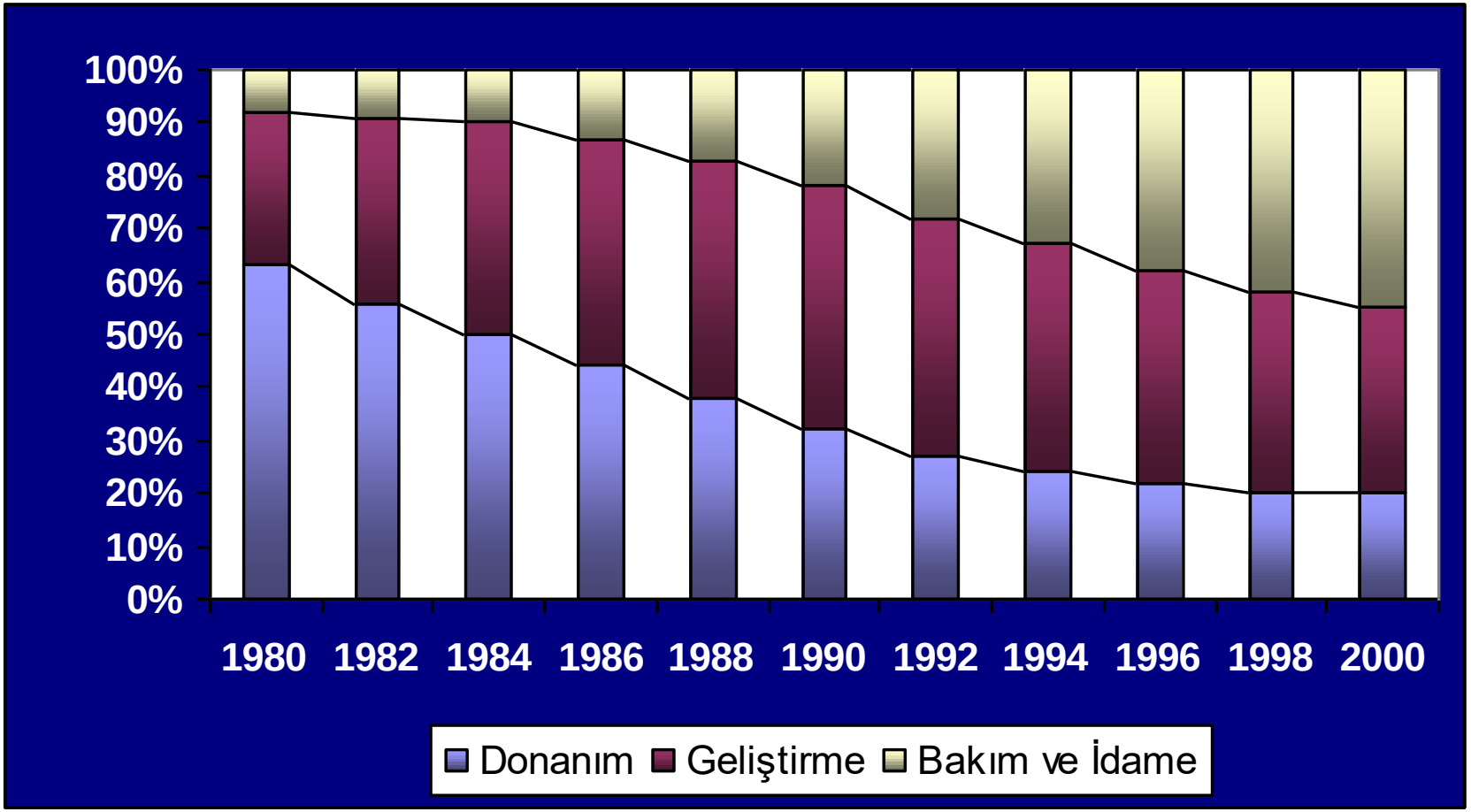
CICS, SQL

5. Kuşak **Yapay Zeka**

Ve Paralel Programlama

CSP, OCCAM

Sistem Harcamaları



Yazılım

Yazılım = Mantık (algoritma) +
Veri (test verisi, bilgi?) +
Belge (dokümanlar) +
İnsan (kullanıcı, geliştirici) +
Program (kod)

- "Bilgisayar sisteminin donanım bileşenleri dışında kalan her şey"

Yazılım

- Mantık, veri, belge, insan ve program bileşenlerinin belirli bir üretim amacına yönelik olarak bir araya getirilmesi, ve

  yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.

Mantık (algoritma)

- Bilgisayarlaştırılmak istenen işin mevcut mantığı yazılıma yansıtılmak durumundadır.
- Bu nedenle mantık (algoritma) bileşeni yazılımın en önemli bileşenlerinden biridir.

Veri

- Her tür yazılım mutlaka bir veri üzerinde çalışmak durumundadır.
- Veri dış ortamdan alınabileceği gibi, yazılım içerisinde de üretilebilir.
- Yazılımın temel amacı “veri”yi “bilgi”ye dönüştürmektir.

Belge (dokümanlar)

- Yazılım üretimi bir mühendislik disiplini gerektirir.
- Mühendislik çalışmalarında izlenen yol ya da kullanılan yaklaşımlar yazılım üretimi için de geçerlidir.
- Yazılım üretimi sırasında, bir çok aşamada yapılan ara üretimlere ait bilgiler (planlama, analiz, tasarım, gerçekleştirim, vb. bilgileri) belli bir düzende belgelenmelidirler.

İnsan (kullanıcı & geliştirici)

- İki boyutludur; yazılımı geliştirenler ve kullananlar.
- Günümüzde artık tek kişi ile yazılım geliştirmekten söz edilmemektedir.
- Yazılım üretimi için bir takım oluşturulmakta ve takımın uyumlu çalışabilmesi için çeşitli yöntemler geliştirilmektedir.

Program (kod)

- Yazılımın ana çıktısı sonuçta bir bilgisayar programıdır.
- Program işletime alındıktan sonra bakım çalışmaları sürekli olarak gündeme gelir.
- Bunun iki temel nedeni:
  - hiç bir program bütünüyle her olasılık göz önüne alınarak test edilemez.
  - işletmeler doğaları gereği dinamik bir yapıya sahiptir ve zaman içerisinde sürekli olarak yeni istek ve gereksinimler ortaya çıkabilmektedir.

Yazılım vs Donanım

- Yazılım geliştirilir vs donanım üretilir. (fabrika ortamında seri üretim)

- Donanım bileşenleri dışarıdan temin edilebilir, ancak yazılımı oluşturan parçalar için bu çoğu zaman mümkün değildir (günümüzde "yeniden kullanılabilir yazılım" %1-2).

Yazılım vs Donanım

- Yazılım eskimez.
- Oysa, her donanımın belli bir ömrü vardır. Ömrünü tamamlayan donanım yenisi ile değiştirilir.
- Yazılımın eskimesi ortaya çıkabilecek yeni ihtiyaçları karşılayamaması, kullandığı teknolojinin eskimesi olarak tanımlanabilir.
- Yeni gereksinimler yazılıma ekler yaparak yansıtılır.

Yazılım vs Donanım

- Yazılım en az donanım kadar önemlidir.
  - Diyaliz makinelerinde kullanılan yazılımların 2000 yılı uyumsuzluğundan ötürü, bir çok diyaliz makinesi çalışamamış ve böbrek hastaları zor durumda kalmıştır.
  - Japonya'da telefon yazılımında ortaya çıkan bir yazılım hatası onbinlerce abonenin saatlerce telefon konuşması yapamamasına neden olmuştur.

Yazılım ve Donanım

- Yazılım kopyalama ve donanım kopyalama farklıdır.
  - Hata toleransı amacıyla, hayati olan bir donanımın sistemde bir kopyası daha bulundurulur ve sistemde biri arızalandığında diğeri çalışmayı devralabilir.
  - Oysa, bir yazılımı sistemde iki ayrı bilgisayar üzerine kopyalamak oluşabilecek hatalara çözüm olmayacaktır. Belki, sisteme aynı işi yapan iki farklı eş yazılım yüklenmesi çözüm olabilir (kritik yazılım sistemleri-uçak avionics).

Tipik Bir Yazılım Üretim Ortamı

- Değişik yetenekte bir çok personel (analist, programcı, test uzmanı, vs.)
- Yazılım çıktısı ile ilgilenen kullanıcılar
- Yeniliğe tepki gösteren kullanıcılar ve yöneticiler !
- Yeterince tanımlanmamış kullanıcı beklentileri
- Personel değişim oranının yüksekliği
- Yüksek eğitim maliyetleri
- Dışsal ve içsel kısıtlar (zaman, maliyet, işgücü, vs)
- Standart ve yöntem eksiklikleri
- Verimsiz kaynak kullanımı
- Mevcut yazılımlardaki kalitesizlik
- Yüksek üretim maliyeti

Yazılım Mühendisliği

- IEEE Tanımı (1993)

"Yazılım Mühendisliği:

Sistemli, düzenli, ölçülebilir bir yaklaşımın

yazılım geliştirmede,

yazılımın işlenilmesinde ve

bakımında uygulanmasıdır.

- *Diğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.*

Yazılım Mühendisliği

- Yazılım üretiminin mühendislik yöntemleriyle yapılmasını öngören ve bu yönde;
  - yöntem,
  - araç
  - teknik ve
  - metodolojiler

  üreten bir disiplindir.

Yazılım Mühendisliği

- Yazılım mühendisliği bir yöntemler, teknikler ve araçlar kümesi olarak değerlendirilebilir.
- Yazılım mühendisliğinin hedefi; yazılım üretimindeki karmaşıklıkları gidermektir.
- Geçmişte kullanılan iş akış şemaları gibi yöntemler günümüzde yetersiz kalmaktadır.
- Ayrıca, yazılım üretimi işi tek kişinin başarabileceği boyuttan çıkmış ve bir takım işi biçimine dönüşmüştür.

Yazılım Mühendisi

- Yazılım Mühendisliği İşini yapan kişidir.
- Temel hedefi; üretimin en az maliyet ve en yüksek nitelikte yapılmasını sağlamaktır.
- Programcı değildir. Ancak programcının tüm yeteneklerine sahiptir.
- Yazılımın daha çok mantıksal boyutuyla ilgilenir ve işi insanlarla ilişkiyi gerektirir.
- Sistem analisti de değildir. Farkı; analist sadece sistemin analiz aşaması ile ilgilenirken, yazılım mühendisi tüm aşamaların içindedir.

Yazılım Hataları

- Bir programı tüm ayrıntıları ile test etmek teorik olarak mümkün olmakla birlikte, uygulamada bu mümkün değildir.
- Yazılım ancak sınırlı sayıda veri ile sınanabilir.

Mantıksal Tasarım	%20
İşlevsel Tasarım	%15
Kodlama	%30
Belgeleme ve Diğerleri	%35

Yazılımda Hata Düzeltme Maliyetleri

- Yazılım üretimindeki hatalar yayılma özelliği gösterir.
- Bu nedenle, hata düzeltme maliyetleri ilerleyen aşamalarda giderek artar.

Analiz	1
Tasarım	5
Kodlama	10
Test	25
Kabul Testi	50
İşletim	100

Yazılım Maliyetleri

- Yazılım = $ 100.000
- Donanım = $ 1000

Yazılım Sistemlerinin Sınıflandırılması

- İşlevlerine göre
- Zamana dayalı özelliklere göre
- Boyuta göre

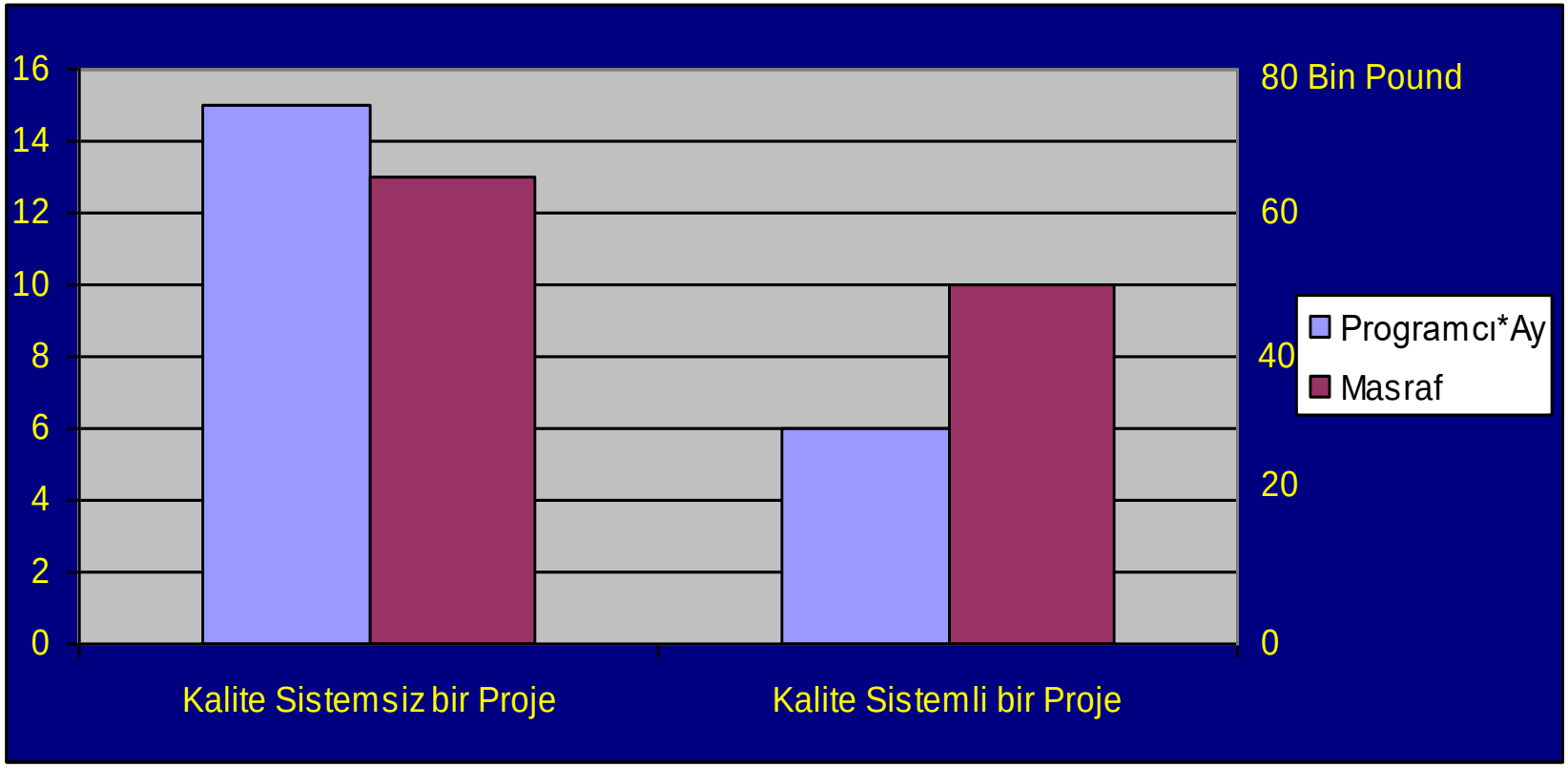
İşleve Göre Sınıflandırma

Hesaplama	Mühendislik Çözümleme
Veri İşleme	Bankacılık
Süreç Temelli	Gömülü Sistemler
Kural Temelli	Robotik, Yapay Zeka
CAD	Sinyal İşleme

Boyuta Göre Sınıflandırma

Küçük (SS<2000)	PC Oyunları Öğrenci Projeleri
Orta (2000<SS<100,000)	CAD BDE Yazılımları
Büyük(100,000<SS<1 Milyon)	İşletim Sistemleri
Çok Büyük (SS>1 Milyon)	Komuta Kontrol Sistemleri Hava Tahmini Sistemleri Yıldız Savaşları Sistemleri

Yazılımda Kalite



Yazılımda Kalite

- Üretim Süreci Boyunca ara ürünlere ilişkin kalite standartlarının geliştirilmesi ve geliştirme işlemlerinin bu standartlara uygunluğunun denetlenmesidir.
- Yazılım kalite sağlama etkinlikleriyle;
  - Yazılım maliyetleri düşürülür,
  - Yazılım üretiminin yönetimi kolaylaşır,
  - Belgeleme ve standart sorunları giderilir.

Yazılımda Kalite

Ekonomi	Tamlık	Yeniden Kullanılabilirlik	Etkinlik	Bütünlük
Güvenirlik	Modülerlik	Belgeleme	Kullanılabilirlik	Temizlik
Değiştirilebilirlik	Geçerlik	Esneklik	Genellik	Sınanabilirlik
Taşınabilirlik	Bakılabilirlik	Anlaşılabilirlik	Birlikte Çalışabilirlik	

Temel terminoloji

- Yazılım (**Software**):
- Yazılım sadece bir bilgisayar programı değildir.
- Basılı veya elektronik ortamdaki her tür dokümanı da içeren ürün.
- Dokümanlar yazılım mühendislerine ve son kullanıcıya yönelik olabilir.
- Uygulama (**Application**) kelimesi de kullanılabilir
- Yazılım ve Donanım adlandırması:
- Yazılım: Software (SW)
- Donanım: **Hardware** (HW)
- İngilizce adlandırma, yazılımın kolaylıkla değiştirilebilecek, oyun hamuru gibi yumuşak bir şeyler olduğu kanısına yöneltir.
- Ancak yazılım daha çok kil veya cam gibidir, bir kere tamamlandıktan sonra değiştirmesi zordur.

Yazılım Türleri

Sistem Yazılımı: System Software

- Diğer programlara hizmet sunmak üzere hazırlanmış programlar.
- Derleyiciler, işletim sistemleri, vb.
- Karmaşık olsa bile belirli, iyi tanımlanmış bilgi yapıları ile uğraşır.
- **Mühendislik Yazılımı / Bilimsel Yazılım : Engineering / Scientific Software**
- Mühendislik ve bilimsel hesaplamalarda kullanılmak üzere hazırlanmış programlar.
- "Numara öğütmek / Number crunching": Bu tip programlar büyük hacimli verilerle uğraştığından bu deyimle karşılaşabilirsiniz.

Yazılım Türleri

Şirket Yazılımı / Kurumsal Uygulamalar (Enterprise software):

- Belirli ticari iş gereksinimlerine yönelik programlar.
- İş süreçleri (business process) ile ilgili bilgiye sahip olmalıdır.
- Genellikle müşteriye özel tasarlanır.
- Veri dönüştürme ve değerlendirme uygulamaları, iş süreçlerinin kimi zaman gerçek zamanlı izlenilmesi, vb.
- Uygulama Yazılımı (Application software):
- Product-line, shrink-wrapped, off-the-shelf, vb.
- Farklı müşteriler tarafından kullanılabilecek genel amaçlı yazılımlar
- Cari hesap uygulamaları, çeşitli otomasyon programları, kelime işlem uygulamaları, vb.

Yazılım Türleri

Gömülü (Embedded) Yazılım :

- Bir ürün veya sistemin bir parçası olup, bu sistemin kendisi ve/veya son kullanıcısı için denetim işlemleri yürüten programlar.
- Gerçek zamanlı (Real Time) uygulamalardır.
- Programın yanıt verme / tepki süresinin (response time) belli bir zaman aşımını (timeout) geçmemesi gerekir.
- Öyle ki, çok hassas bir yanıtın geç gelmesi yerine, yeterli bir yanıtın çabuk gelmesi daha önemli olabilir.

Yazılım Türleri

Ağ Uygulamaları (Web applications):

- Ağ üzerinden haberleşerek hizmet almaya veya vermeye yönelik uygulamalar.
- Şirket yazılımları ile etkileşimde bulunabilirler.
- E-ticaret, B2B, B2C, web servisleri, web tarayıcıları, vb.
- B2B: Business to Business
- İki veya daha fazla ticari firma arasında çalışan uygulamalar.
- Birden çoğa, çoktan çoğa, işbirliğine ve ticari işlemlere yönelik
çeşitli uygulamalardır.
- B2C: Business to Customer
- Doğrudan son kullanıcıya satış amaçlı.

Yazılım Türleri

Yapay Zeka (Artificial Intelligence: AI) Yazılımları :

- Sayısal olmayan algoritmalarla karmaşık sorunları çözmeye yönelik yazılımlar.

- Robotik, uzman sistemler (expert systems), örüntü tanıma (pattern recognition) (ses ve görüntü), vb.

Yazılım Türleri

Eski Yazılım (Legacy Software)

- İş sürecinin önemli bir parçası olan ve çok uzun süredir kullanılan yazılımlar.
- Şirketler, yazılım sistemleri dahil, yaptıkları yatırımı mümkün olan en uzun sürece kullanmak ister.
- Ancak iş alanındaki gereksinimler hızla değişebilir.
- Yazılım artık yeni ihtiyaçları karşılayacak şekilde esnetilemiyorsa, yazılım yeniden tasarlanmalıdır.
- Eski uygulamaya şirketin diğer bir çok süreci ve bilgi sistemi bağımlı ise, tümleştirme (integration) çalışmaları zor olabilir.

Yazılım Yaşam Döngüsü

Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen aşamalardır.

- Döngü: Kullanım dışı bırakılan yazılımın yerine yenisi hazırlanabilir.
- Döngünün aşamalarının belirlenmesi ve tanımlanması ile yazılım geliştirme modelleri/süreçleri elde edilir.

Yazılım Geliştirme Süreçleri / Modelleri

Yazılım geliştirme bir süreçtir (**sw development** process)

- Süreç: Önceden belirlenmiş adımlardan oluşan iş akışı.
- Yazılım geliştirme modelleri, sürecin yapısını ve adımlarını belirler.

Yazılım Geliştirme Süreçleri

Modellerin tanımladığı adımlar arasında farklar olmakla beraber, her süreç modelinde bulunan genel işlemler şu şekildedir:

- Çözümleme (Analysis)
- Ne yapılacak?
- Tasarım (Design)
- Nasıl yapılacak?
- Gerçekleme (Implementation)
- Haydi yap!
- Sınama (Testing)
- Doğru yaptın mı?
- Bakım (Maintenance)
- Değişmeyen tek şey değişimin kendisidir!

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği

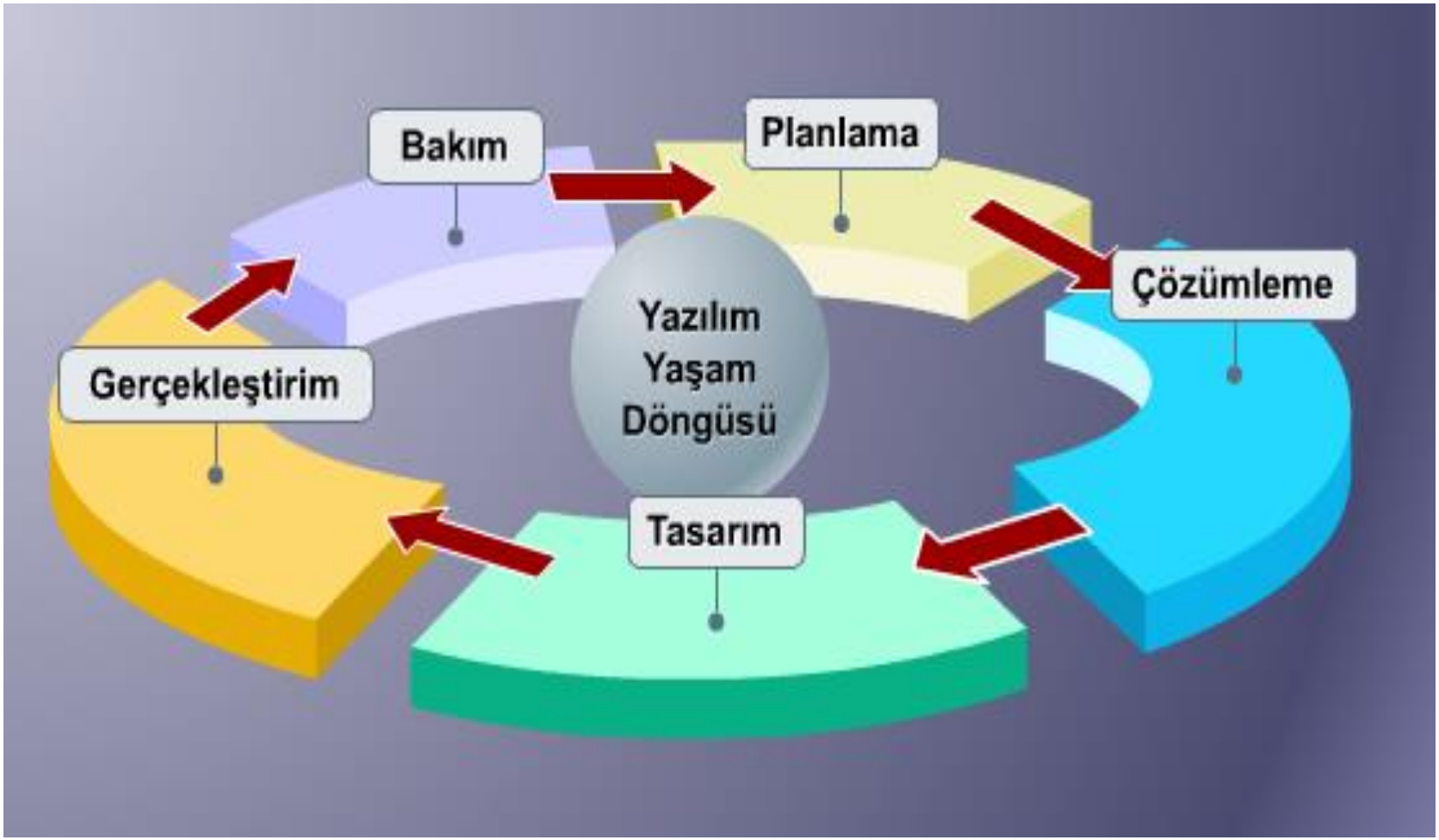
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ-Hafta 2

Hedef

- Bu bölümde, yazılım geliştirme yaşam döngüsünün çekirdek aşamaları olan, **planlama**, **çözümleme**, **tasarım**, **gerçekleştirim** ve **bakım** aşamaları tanıtılmakta, yaşam döngüsü modelleri ya da süreç modelleri olarak da **yaşam döngüsü**nün (life cycle) uygulamada kullanım modellerinin açıklanması hedeflenmektedir.
- Metododoloji, kavramı bu bölümde açıklanmaktadır.

Yazılım Yaşam Döngüsü

- Planlama, çözümleme, tasarım, gerçekleştirim, bakım

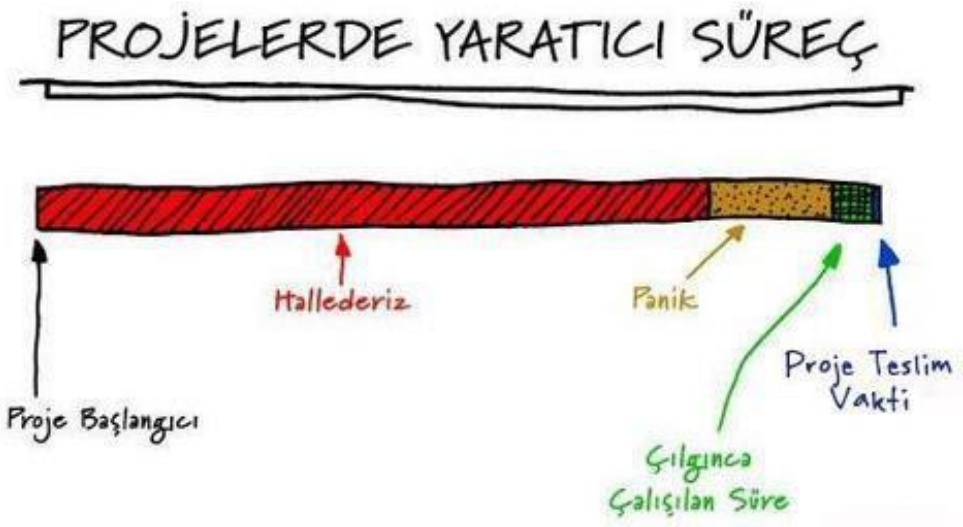


Yazılım Yaşam Döngüsü

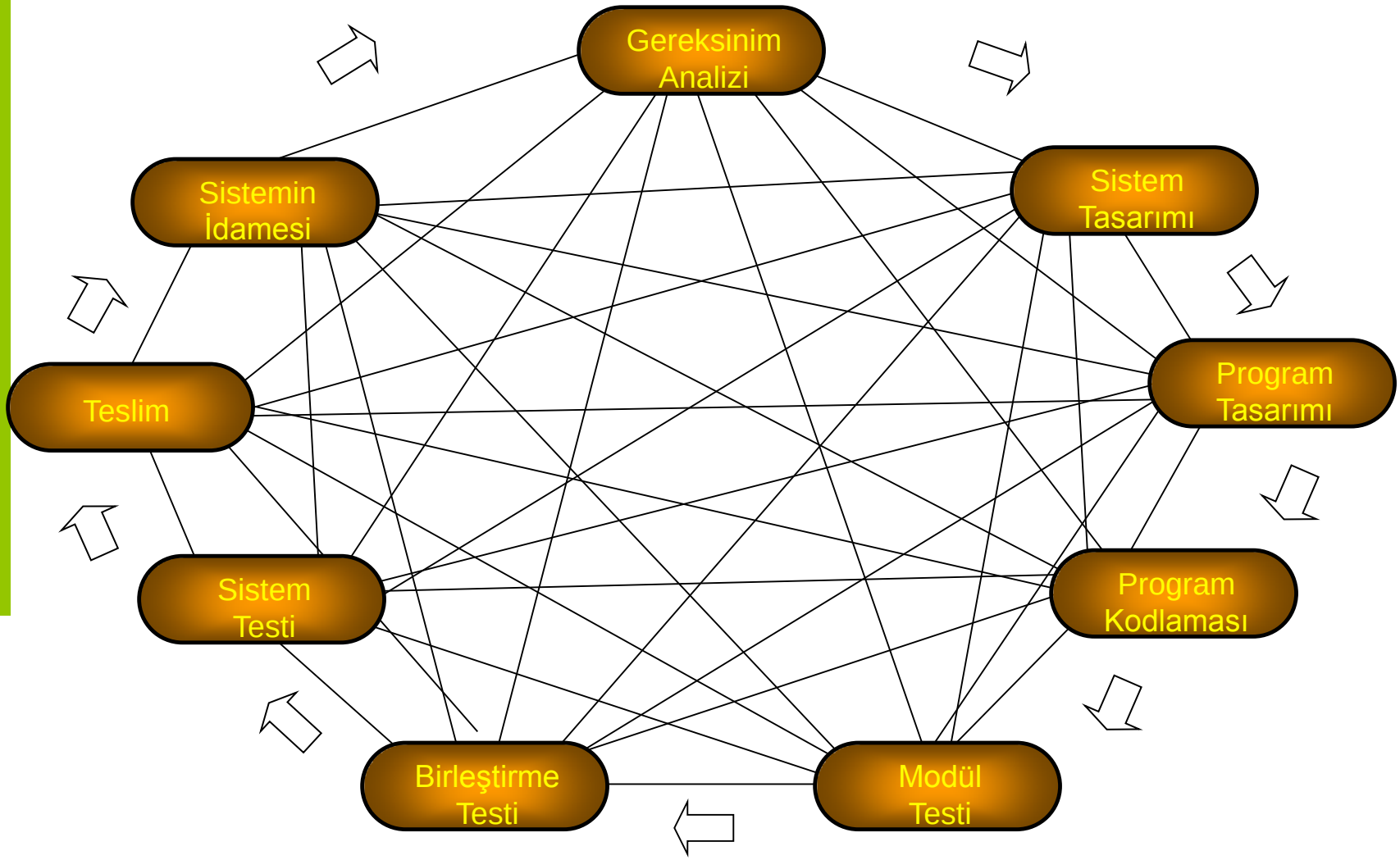
- Herhangi bir yazılımın, **üretim aşaması** ve **kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
- Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
- Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.
- Bu nedenle yazılım yaşam döngüsünün tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü-Ne sağlar?

- Mühendislik faaliyetlerinin (üretim, işletme ve bakım aşamaları) yazılıma uyarlanması
- Üretimden kullanıma kadar tüm sürecin fazlara ayrılması ve böylece yönetim kolaylığı
- Her bir aşamada ne yapılacak?
- Her bir aşamadaki standartlara uyumluluk ile kaliteli yazılım geliştirme vb.



Gerçek Hayatta Program Geliştirme



Yazılım Yaşam Döngüsü Temel Adımları

Planlama: Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite (olurluk-yapılabilirlik) çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

Analiz: Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Mevcut var olan işler incelenir, temel sorunlar ortaya çıkarılır.

Tasarım: Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.

mantıksal; önerilen sistemin yapısı anlatılır,olası örgütsel değişiklikler önerilir.

fiziksel; yazılımı içeren bileşenler ve bunların ayrıntıları.

Gerçekleştirim: Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.

Bakım: Hata giderme ve yeni eklentiler yapma aşaması (teslimden sonra).

Yazılım Yaşam Döngüsü Temel Adımları

- Yaşam döngüsünün temel adımları çekirdek süreçler (core processes) olarak da adlandırılır.

- Bu süreçlerin gerçekleştirilmesi amacıyla
  - belirtim (specification) yöntemleri - bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler
  - Süreç (process) modelleri - yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlayan modeller

kullanılır.

Belirtim Yöntemleri

- Süreç Akışı İçin Kullanılan Belirtim Yöntemleri

  Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler (Veri Akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları).

- Süreç Tanımlama Yöntemleri

  Süreçlerin iç işleyişini göstermek için kullanılan yöntemler (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).

- Veri Tanımlama Yöntemleri

  Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemler (Nesne İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü).

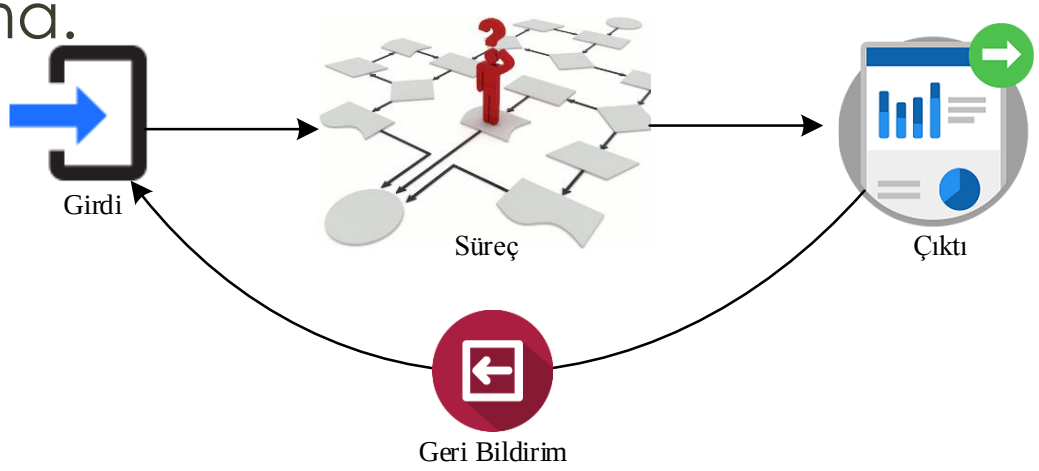
Yazılım Süreci Modelleri

- Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir.
- Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.

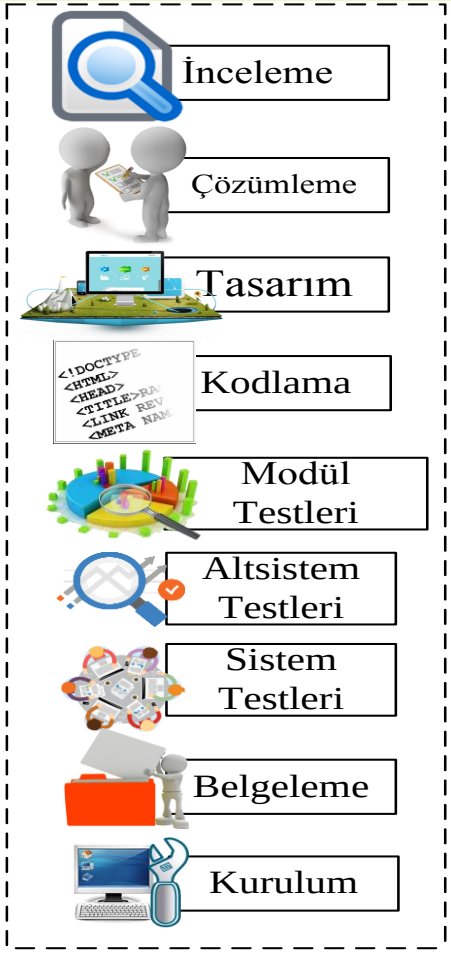
1. Gelişigüzel Model
2. Barok Modeli
3. Çağlayan (Şelale) Modeli
4. V Modeli
5. Helezonik (Spiral) Model
6. Evrimsel Model
7. Artırımsal Model
8. Araştırma Tabanlı Model

Gelişigüzel Model

- Herhangi bir model ya da yöntem yok.
- Geliştiren kişiye bağımlı (belli bir süre sonra o kişi bile sistemi anlayamaz hale gelebilir ve geliştirme güçlüğü yaşar).
- İzlenebilirliği ve bakımı oldukça zor.
- 60'lı yıllarda kullanılan ilkel model
- Genellikle tek kişilik üretim ortamı.
- Basit programlama.



Barok Modeli



Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.

70'li yıllar.

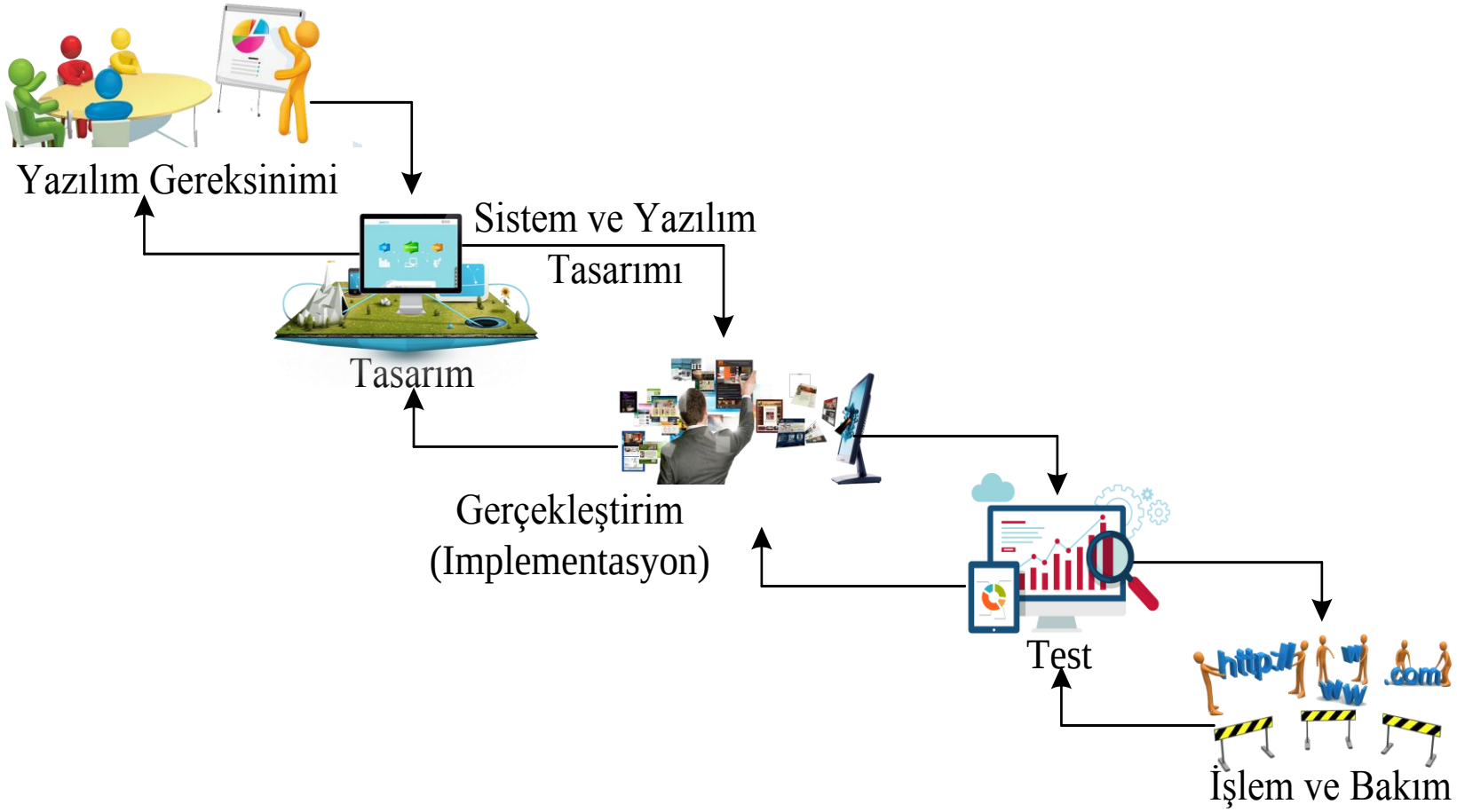
Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasını öngörür.

Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.

Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.

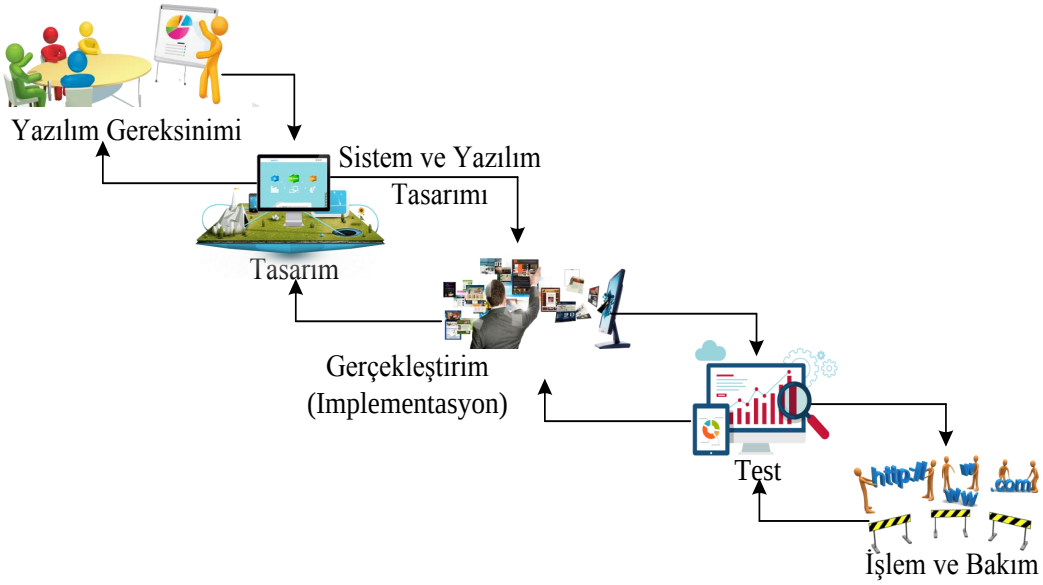
Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir

Çağlayan (Şelale) Modeli



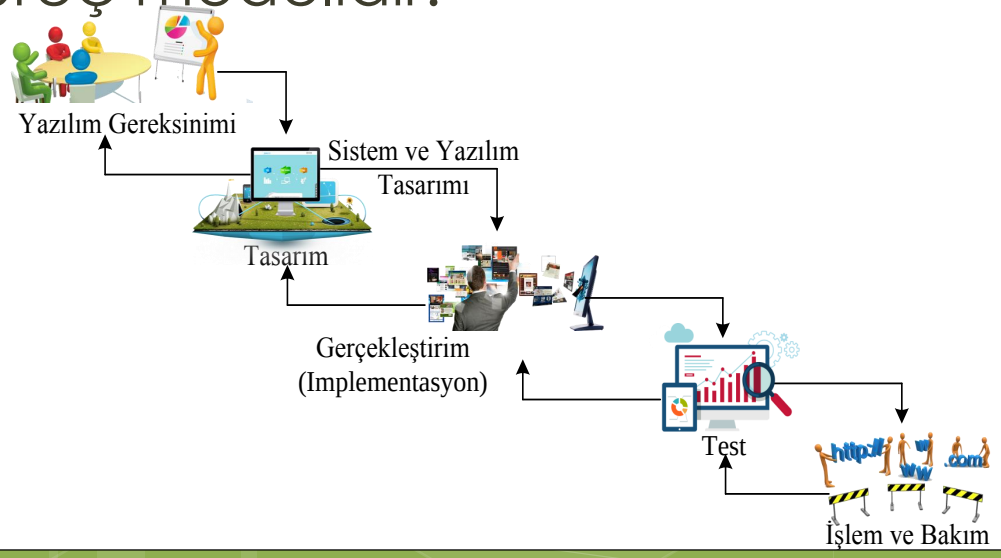
Çağlayan (Şelale) Modeli

- Yaşam döngüsü temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir.
- İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
- Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.



Çağlayan (Şelale) Modeli

- Barok modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.
- Barok modele göre geri dönüşler iyi tanımlanmıştır.
- Yazılım tanımlamada belirsizlik yok (ya da az) ise ve yazılım üretimi çok zaman almayacak ise uygun bir süreç modelidir.



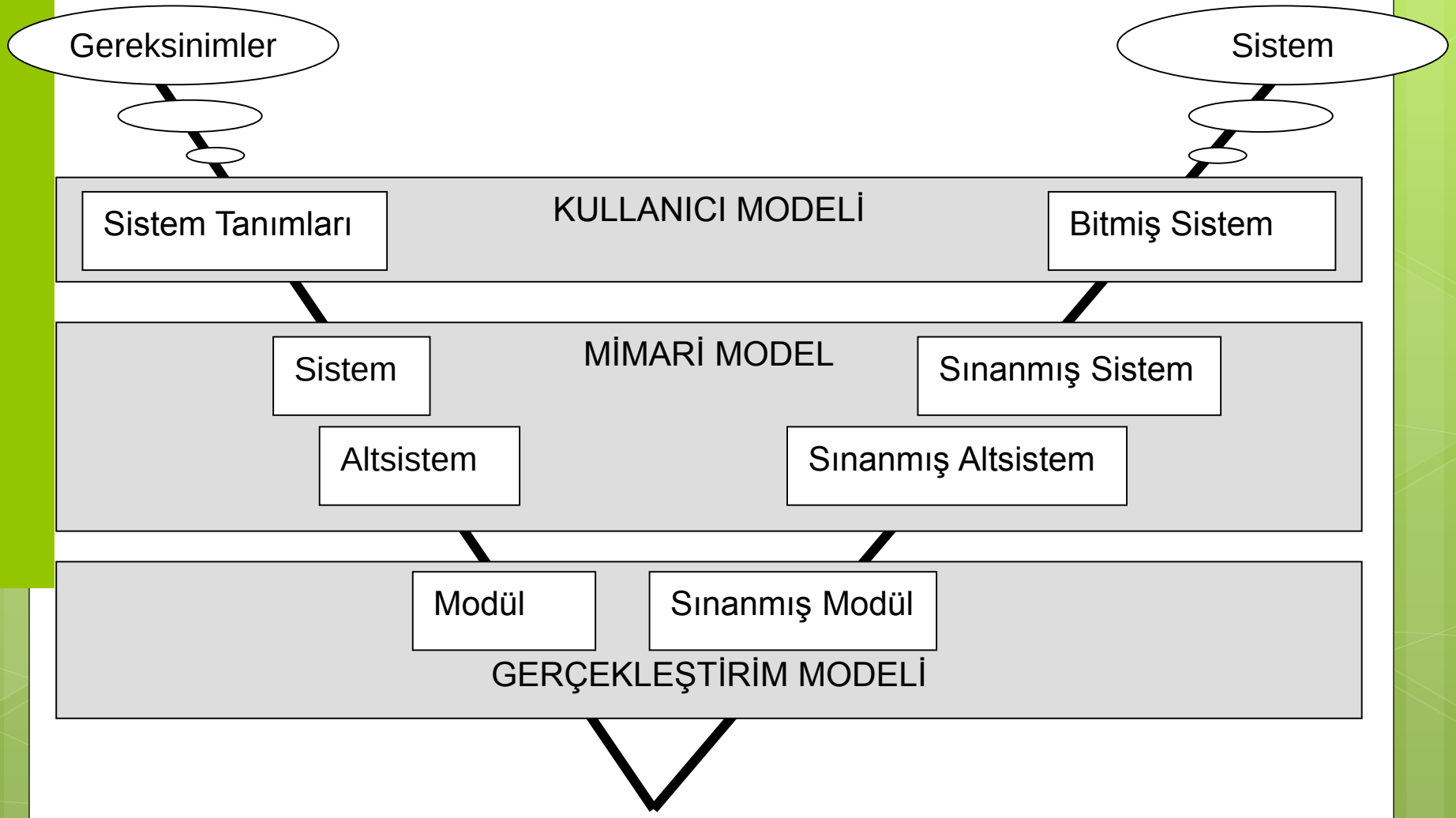
Avantajları

- Müşteriler ve son kullanıcılar tarafından da iyi bilinen anlaşılabilen adımlardan oluşur.
- İterasyonlar (tekrarlamalar) bir sonraki ve bir önceki adımlarla gerçekleşir, daha uzak adımlarla olması nadirdir.
- Değişiklik süreci yönetilebilir birimlere bölünmüştür.
- Gereksinim adımı tamamlandıktan sonra sağlam bir temel oluşur. Aşamaları iyi anlaşılabilir.
- Proje yöneticileri için işin dağılımını yapma açısından kolaydır.
- Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
- Kalite gereksinimlerinin bütçe ve zaman kısıtlamasına göre çok daha önemli olduğu projelerde iyi çalışır.

Sorunları

- Gerçek yaşamdaki projeler genelde yineleme gerektirir.
- Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.
- Gereksinim tanımlamaları çoğu kez net bir şekilde yapılamadığından dolayı, yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.
- Yazılım üretim ekipleri bir an önce program yazma, çalıştırma ve sonucu görme eğiliminde olduklarından, bu model ile yapılan üretimlerde ekip mutsuzlaşmakta ve kod yazma dışında kalan (ve iş yükünün %80'ini içeren) kesime önem vermemektedirler.
- Üst düzey yönetimlerin ürünü görme süresinin uzun oluşu, projenin bitmeyeceği ve sürekli gider merkezi haline geldiği düşüncesini yaygınlaştırmaktadır.

V Süreç Modeli



V Süreç Modeli

Sol taraf üretim, sağ taraf sınama işlemleridir.

V süreç modelinin temel çıktıları;

- Kullanıcı Modeli

  Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

- Mimari Model

  Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

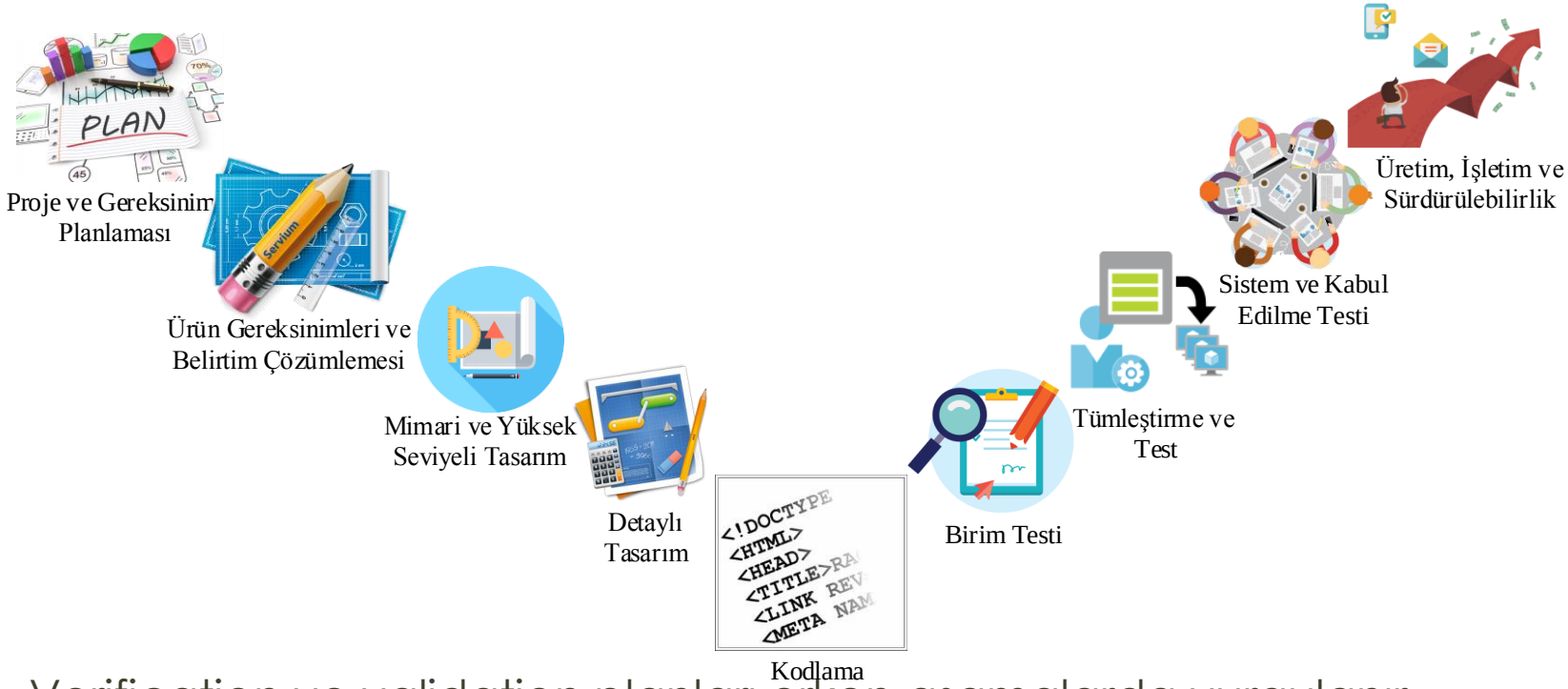
- Gerçekleştirim Modeli

  Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

V Süreç Modeli

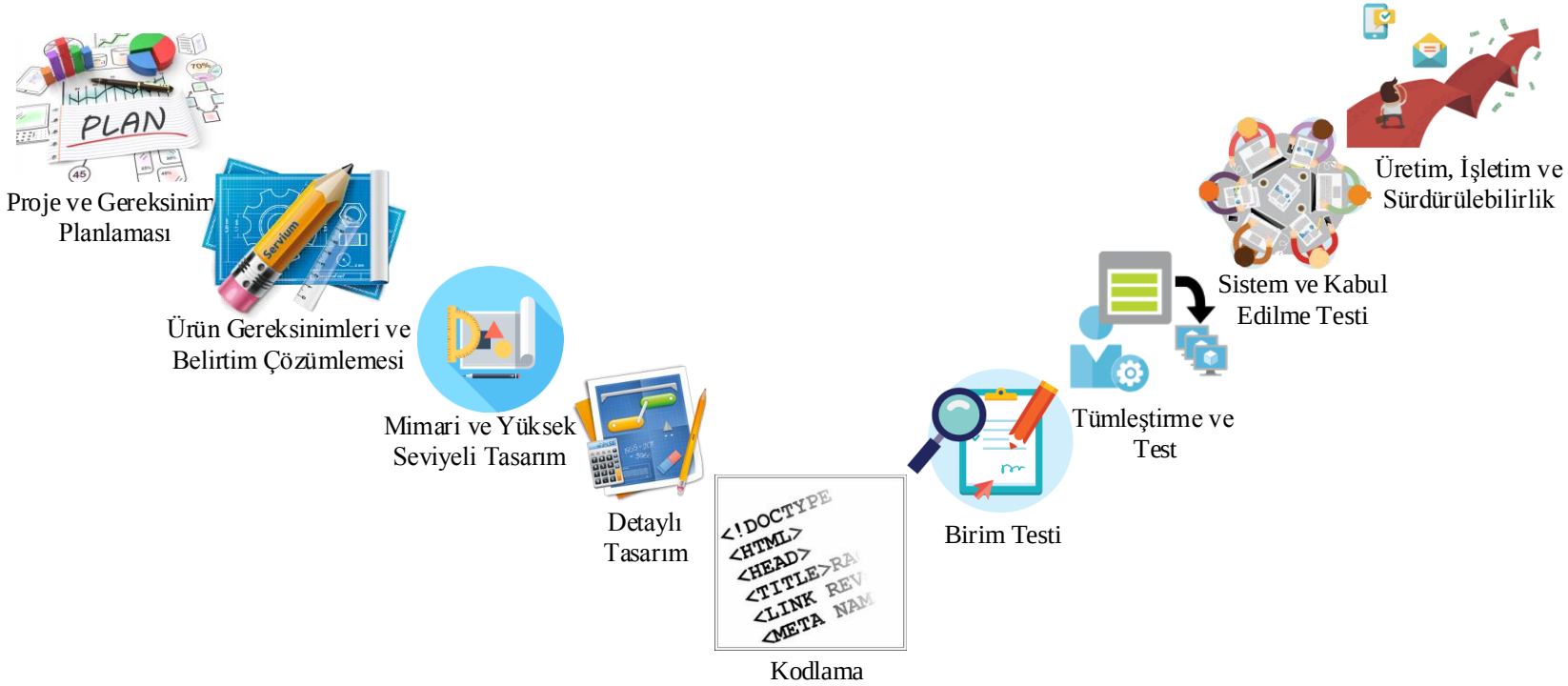
- Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygun bir modeldir.
- Model, kullanıcının projeye katkısını arttırmaktadır.
- BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur:
  - İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta,
  - İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçeklenmektedir.

V Modeli-Avantajlar



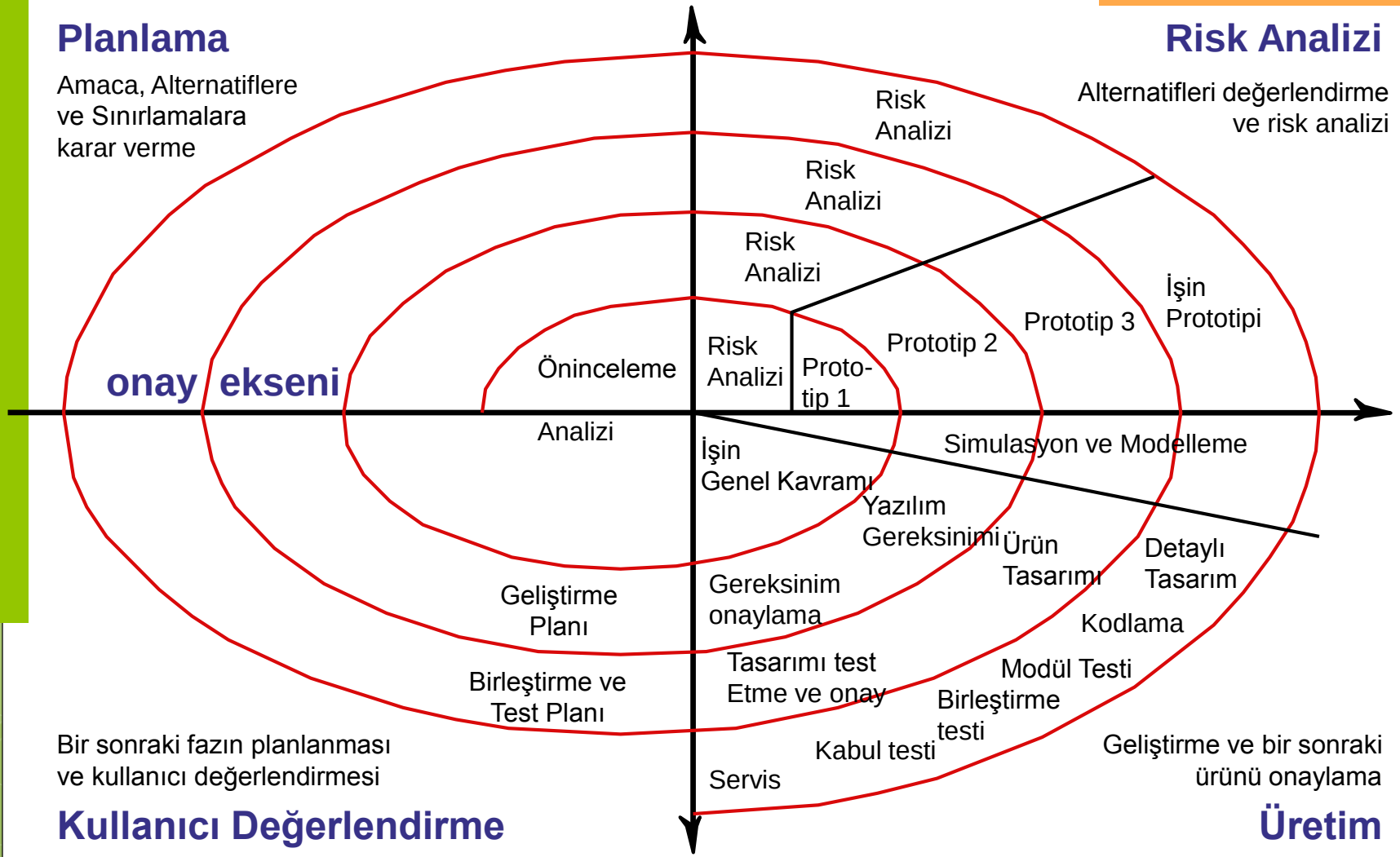
- Verification ve validation planları erken aşamalarda vurgulanır.
- Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
- Proje yönetimi tarafında takibi kolaydır
- Kullanımı kolaydır.

V Modeli-Dezavantajlar



- Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz.
- Aşamalar arasında tekrarlamaları kullanmaz.
- Risk çözümleme ile ilgili aktiviteleri içermez.

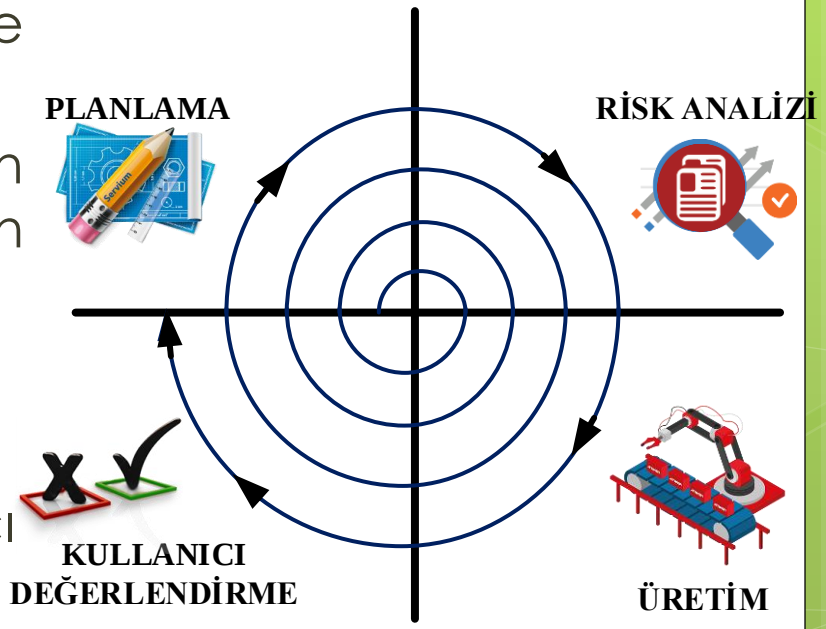
Helezonik(Spiral) Modeli



Helezonik Model

1. **Planlama:** Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme
2. **Risk Analizi:** Risk seçeneklerinin araştırılması ve risklerin belirlenmesi
3. **Üretim:** Ara ürünün üretilmesi
4. **Kullanıcı Değerlendirmesi**

   Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler



Helezonik 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.

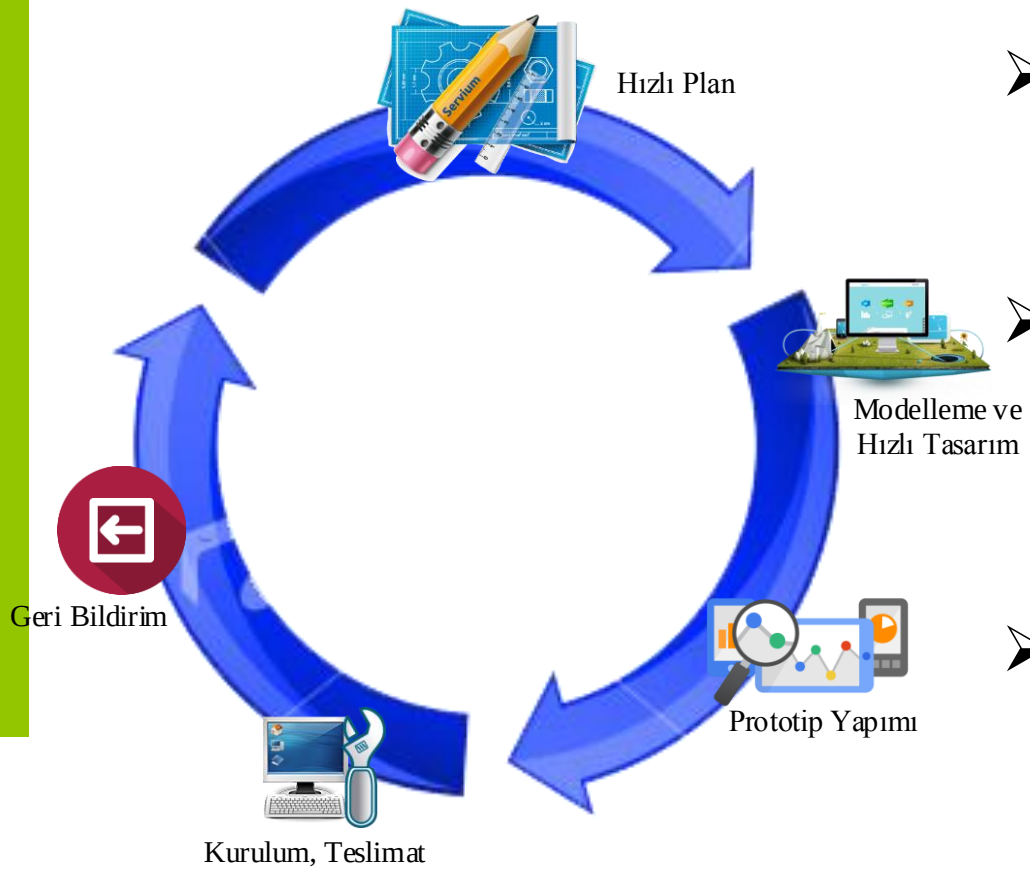
3. Yazılım Geliştirici (Mühendis) Bakışı

   Yazılımın kodlanması ve sınanması daha erken başlar.

Helezonik Model

- 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.

Prototipleme



- Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.
- Gereksinimler netleştikçe prototip kullanıcı ihtiyaçlarına göre düzeltilir
- Müşteri memnun olana kadar düzeltmelere devam et

Prototipleme-Avantajları

- Kullanıcı sistem gereksinimlerini görebilir
- Karmaşa ve yanlış anlaşılmaları engeller
- Yeni ve beklenmeyen gereksinimler netleştirilebilir
- Risk kontrolü sağlanır

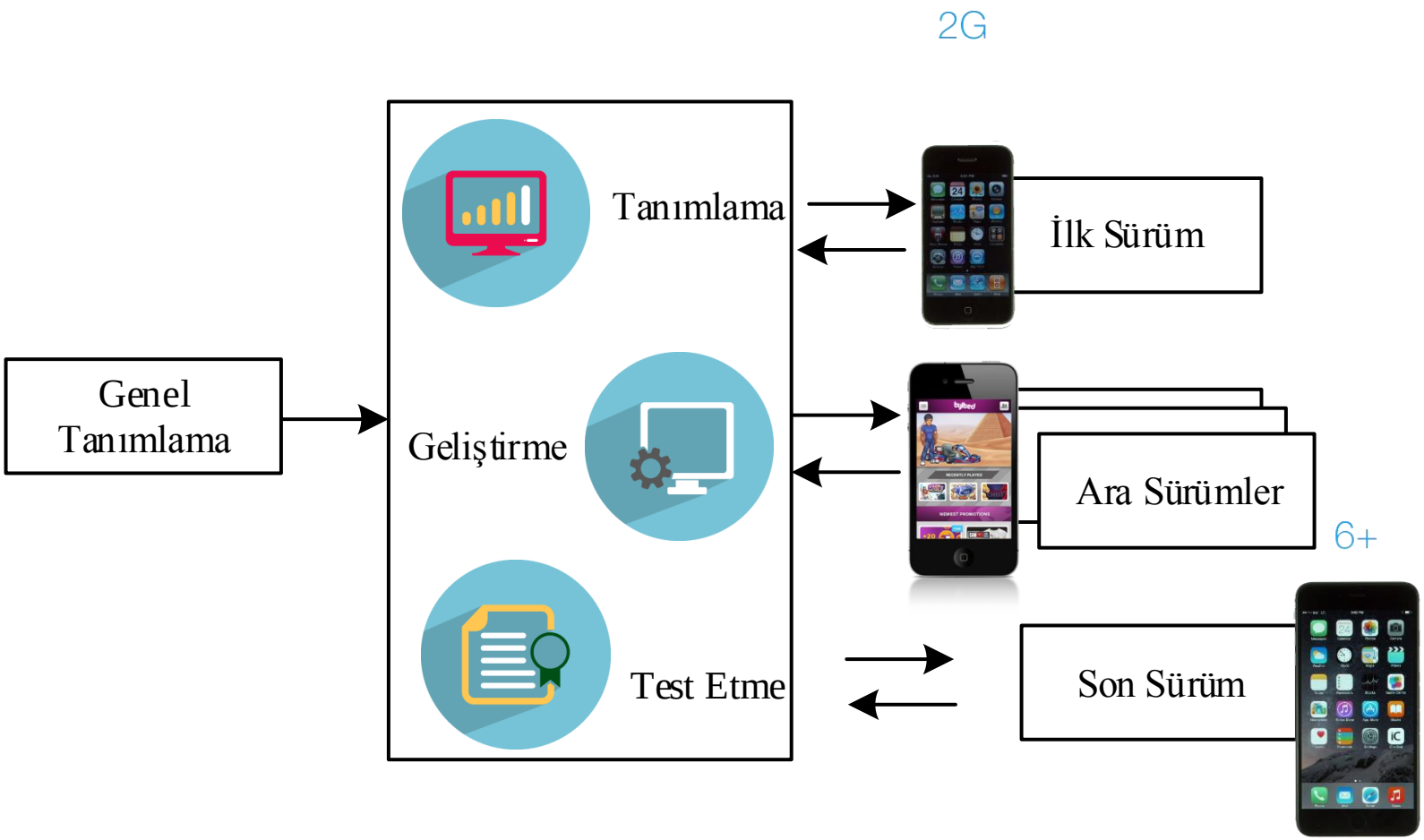
Prototipleme-Dezavantajları

- Belgelendirmesi olmayan hızlı ve kirli (quick and dirty) prototipler
- Prototip hedefleri net değilse kod hackleme ya da jenga başlar
- Düzeltme aşaması atlanırsa, düşük performansa yol açar
- Müşteri prototipten de son ürün gibi görünüm ve etki bekler

Evrimsel Geliştirme Süreç Modeli

- 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.

Evrimsel Geliştirme Süreç Modeli



Örnek

- Çok birimli banka uygulamaları.
- Önce sistem geliştirilir ve Şube-1'e yüklenir.
- Daha sonra aksaklıklar giderilerek geliştirilen sistem Şube-2'ye yüklenir.
- Daha sonra geliştirilen sistem Şube-3'e,…. yüklenir.
- Belirli aralıklarla eski şubelerdeki güncellemeler yapılır.

Evrimsel Geliştirme Süreç Modeli

İki çeşit evrimsel geliştirme vardır:

Keşifçi geliştirme (exploratory development)

- Hedef: Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmek
- İyi anlaşılan gereksinimlerle başlanmalıdır.

“ Ne istediğimi sana söyleyemem ama onu gördüğümde bilirim”

Atılacak prototipleme (throw-away prototyping)

- Hedef: Sistem gereksinimlerini anlamak
- Tam anlaşılmamış gereksinimlerle başlar

Aksaklığı

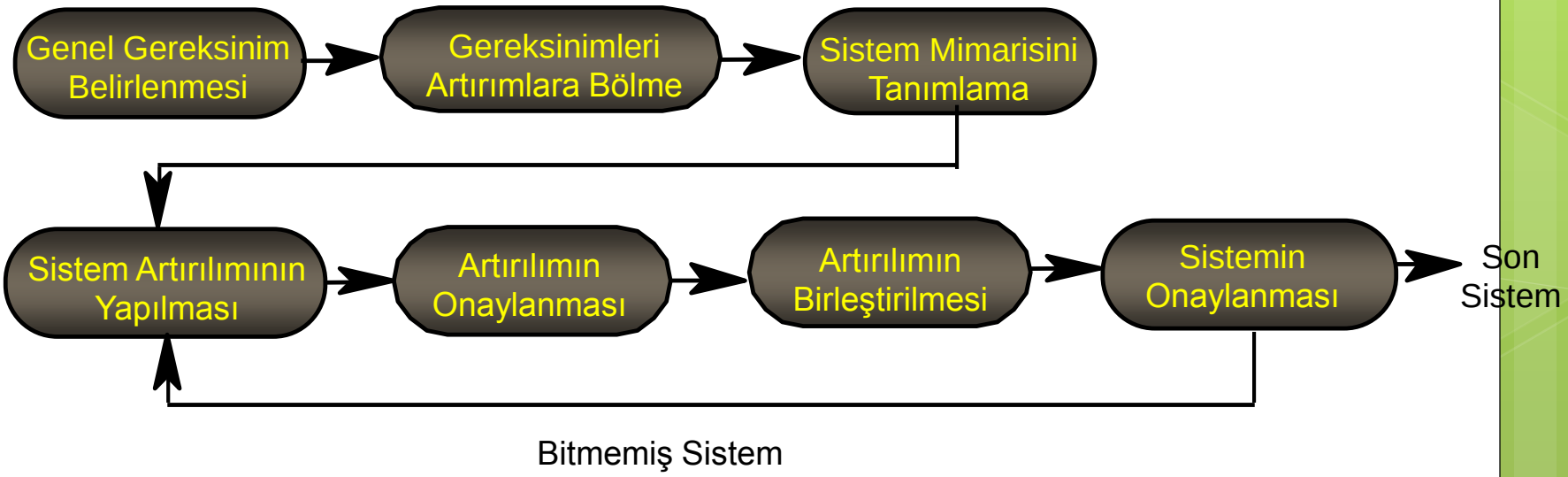
- Değişiklik denetimi
- Konfigürasyon Yönetimidir
  - Sürüm Yönetimi
  - Değişiklik Yönetimi
  - Kalite Yönetimi

Artırımsal Geliştirme Süreç Modeli

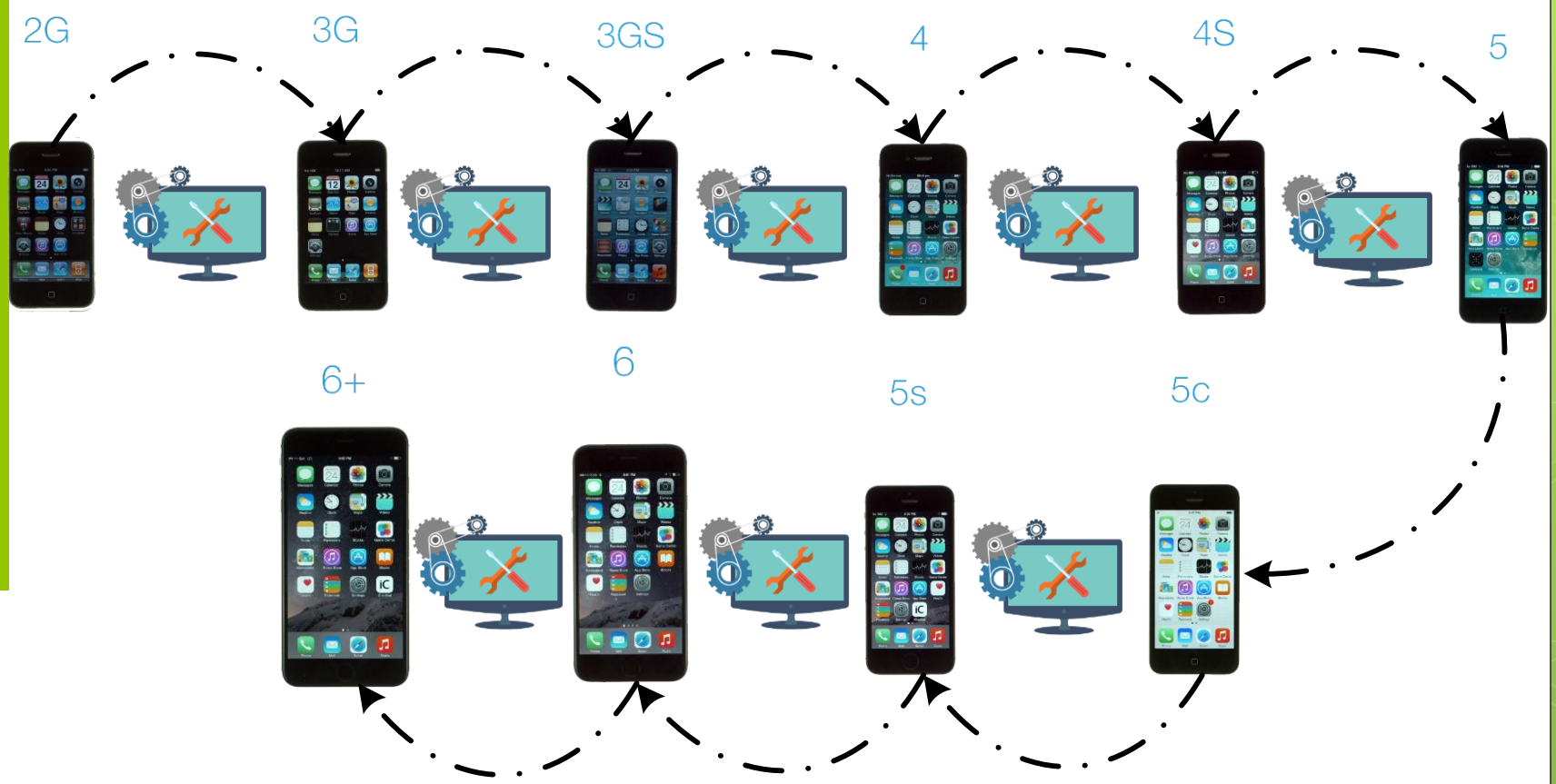
- Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.
- Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 2 haftada bir gelişiminin izlenmesi (bitirme tezleri).
- Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.
- Bir taraftan kullanım, diğer taraftan üretim yapılır.

Artırımsal Geliştirme Süreç Modeli

Artırımsal Geliştirme Süreç Modeli



Artırımsal Geliştirme Süreç Modeli



Araştırma Tabanlı Süreç Modeli

- Yap-at prototipi olarak ta bilinir.
- Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.
- Yapılan 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.

Araştırma Tabanlı Süreç Modeli

Örnek

- En Hızlı Çalışan asal sayı test programı!
- En Büyük asal sayıyı bulma programı!
- Satranç programı!

Metodolojiler

- Metodoloji: Bir BT projesi ya da yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütünü.
- Bir metodoloji,
  - bir süreç modelini ve
  - belirli sayıda belirtim yöntemini içerir
- Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır

Bir Metodolojide Bulunması Gereken Temel Bileşenler (Özellikler)

- Ayrıntılandırılmış bir süreç modeli
- Ayrıntılı süreç tanımları
- İyi tanımlı üretim yöntemleri
- Süreçlerarası arayüz 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

Bir Metodolojide Bulunması Gereken Temel Bileşenler

- Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (IEEE, ISO, vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.
- Kullanılan süreç modelleri ve belirtim yöntemleri zaman içinde değiştiği için standart ve rehberler de sürekli güncellenmektedir.
- Bir kuruluşun kendi metodolojisini geliştirmesi oldukça kapsamlı, zaman alıcı ve uzmanlık gerektiren bir faaliyet olup, istatistikler yaklaşık 50 kişi/ay'lık bir iş gücü gerektirdiğini göstermektedir.

Bir Metodoloji Örneği

- Yourdan Yapısal Sistem Tasarımı Metodolojisi.
- Kolay uygulanabilir bir model olup, günümüzde oldukça yaygın olarak kullanılmaktadır.
- Çağlayan modelini temel almaktadır.
- Bir çok CASE aracı tarafından doğrudan desteklenmektedir.

Yourdon Yapısal Sistem Tasarım Metodolojisi

Aşama	Kullanılan Yöntem ve Araçlar	Ne için Kullanıldığı	Çıktı
Planlama	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Maliyet Kestirim Yöntemi, Proje Yönetim Araçları	Süreç İnceleme Kaynak Kestirimi Proje Yönetimi	Proje Planı
Analiz	Veri Akış Şemaları,Süreç Belirtimleri,Görüşme,Nesne ilişki şemaları ,Veri	Süreç Analizi Veri Analizi	Sistem Analiz Raporu
Analizden Tasarıma Geçiş	Akışa Dayalı Analiz, Süreç belirtimlerinin program tasarım diline dönüştürülmesi, Nesne ilişki şemalarının veri tablosuna dönüştürülmesi	Başlangıç Tasarımı Ayrıntılı Tasarım Başlangıç Veri tasarımı	Başlangıç Tasarım Raporu
Tasarım	Yapısal Şemalar, Program Tasarım Dili, Veri Tabanı Tabloları	Genel Tasarım Ayrıntılı Tasarım Veri Tasarımı	Sistem Tasarım Raporu

Süreç Modeli	Örnek Proje
1. Gelişigüzel Model	Öğrenci Ders Projeleri
2. Barok Model	Kullanım Kılavuzu Gerektiren Basit Projeler
3. Çağlayan Modeli	Hastane Sistemleri, Kontrol Sistemleri
4. V Modeli	CRM (Müşteri İlişkileri Yönetimi) Projeleri
5. Helezonik Model	Askeri Savunma Yazılım Sistemleri
6. Arttırımsal Geliştirme	Öğrenci Bitirme Projeleri
7. Prototipleme	Yüksek Maliyetli Projelerin İlk Aşamasının Bilgisayar Destekli Gerçekleştirilmesi
8. Evrimsel Geliştirme	Çok Birimli Organizasyonlar İçin (Bankacılık Uygulamaları)
9. Araştırma Tabanlı Model	En Hızlı Çalışan Asal Sayı Test Programı, En Büyük Asal Sayıyı Bulma Programı, Satranç Programı

YMH 114 Stratejik Bakış

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

YMH 114 için Stratejik Bakış-Sunum İçeriği

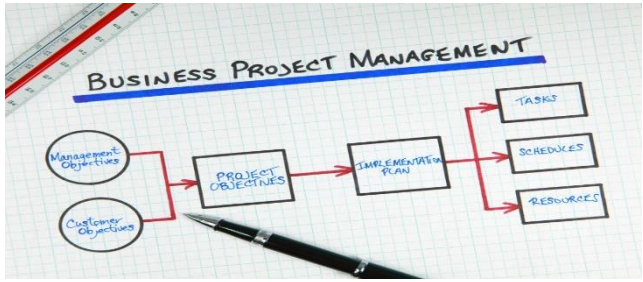
- Proje Nedir?
- Proje Neleri Sağlamalıdır?
- Proje-Fikir İlişkisi: Her Fikir Proje midir?
- Amaç Nedir? Amaç neleri içermelidir?
- Sorun Analizi ve Hedef Analizi ne demektir?
- Araştırma Projesi Nedir?
- Proje seçiminde öneriler?

Proje Nedir?

“Proje” belirli bir yerde, belirli bir zaman ve bütçe çerçevesinde, bir başlama ve bitiş noktasına sahip, önceden belirlenmiş amaçlara ulaşılmasını sağlayacak olan faaliyetler topluluğudur.

Proje Neleri Sağlamalıdır?

- Belirli bir yerde uygulanmalı,
- Takvimi olmalı,
- Bütçesi olmalı,
- Tanımlanmış faaliyetleri olmalıdır.



Proje-Fikir İlişkisi: Her Fikir Proje midir?

Her fikir bir proje midir?

Hayır, değildir. Bir fikrin proje olabilmesi için aşağıdaki sorulara cevap verebilecek olgunluğa erişmiş olması gerekmektedir.

- ♦ Nerede uygulayacaksınız?
- ♦ Ne kadar sürede bitireceksiniz?
- ♦ Belirlediğiniz yer ve sürede gerçekleştireceğiniz faaliyetler için ne kadar mali kaynağa ihtiyacınız var?
- ♦ Belirlediğiniz yerde ve sürede hangi faaliyetleri gerçekleştireceksiniz?

Amaç Nedir? Amaç neleri içermelidir?

- Amaçlarınız projeniz ile ortadan kaldırmayı, iyileştirmeyi önerdiğiniz sorunlara işaret etmelidir. Dolayısı ile projenin genel amacı ve özel amacını/amaçlarını doğru ve detaylı bir şekilde ortaya koyabilmek, tekrarlara düşmemek adına **sorun analizi** ve **hedef analizi** yapmanız önerilir.

Sorun Analizi ve Hedef Analizi ne demektir?

Sorun analizi, projenin çözmekte rol oynayacağı/katkı sağlayacağı temel sorunun belirlenmesi ve bu soruna bağlı alt sorunların da tanımlanmasıdır.

Hedef analizi ise, sorun analizinde belirlenen temel sorunu ve alt sorunları olumluya çeviren önermeler ortaya koymaktır.

‘Amaç’ kısmının yazılması

- Bu açıklamalar doğrultusunda, denilebilir ki projenin genel amacı, **temel sorunun** çözümüne yönelik bir **hedef**, özel amaç(lar) ise, alt sorun(lar)un çözümüne yönelik hedef(ler)tir. Özel amaçların, proje sonunda ölçülebilir, ulaşılabilir ve gerçekçi özellikte olması gerekir ve başvuru sahibi, bu amaçların gerçekleştiğini proje sonunda **ispat** etmekle yükümlüdür.

Amaç kısmının yazılması

- Özel amaçlar faaliyetler ile karıştırılmamalıdır. **Örneğin;** Bir eğitimin gerçekleşmesi, bir makinenin alınması özel amaç değil, faaliyettir.
- Eğitim sonucunda katılımcıların teknik kapasite, bilgi ve becerilerinin artırılması, alınan makine ile üretim miktarında artış sağlamak ise faaliyetlerinize bağlı olarak ulaşabileceğiniz özel amaçlardır.

1.1 AMAÇ

Web sunucuları veya aktif ağ cihazları tarafından farklı biçimlerde saklanan web erişim kütüklerinin temizlenmesine yönelik uygun bir yazılım geliştirmektir. Ayrıca, geliştirilen bu yazılım ile temizlenen erişim kütüklerinden anlamlı bilgilerin elde edilmesine yönelik analiz yapılmıştır. Yapılan bu analiz, web sitelerine erişen kullanıcıların davranışları hakkında bilgi edinilmesini ve elde edilen bu bilgilerle kullanıcı davranışlarının tahmin edilmesine olanak sağlamaktadır. Bu geliştirilen uygulamaya ilave olarak ilerleyen aşamalarda diğer saldırı türlerinin tespiti ve web sunucularına entegre olarak anlık tespit yapılması da projenin amaçlarındandır.

Araştırma Projesi

Araştırma projesi deneyimi, öğrenciye araştırma metodolojisini öğrenme ve danışmanıyla yakın bir çalışma olanağı sunar.

Genellikle ileri düzey kavramların, çeşitli deneysel tekniklerin kullanılmasını gerektirir. Literatür taraması tek başına bir araştırma projesi değildir, projede bulunması gereken en önemli unsurlardan biridir.

Araştırma Süreci

Bir konu hakkında araştırma yapılmak istendiğinde öncelikle aşağıda yer alan araştırma süreçlerinden sırasıyla geçilmesi gerekmektedir:

i. Öncelikle bir literatür taraması yapılmalı,
ii. Bu alandaki çalışmalar iyice okunmalı ve notlar çıkarılmalı,
iii. Literatür taraması sonunda araştırma sorusu belirlenmeli,
iv. Analiz yöntemi belirlenmeli,
v. Veriler toplanmalı,
vi. Analiz yapılmalı,
vii. Analiz sonucunda elde edilen bulgular yorumlanmalı,
viii. Son olarak proje raporu yazılmalıdır.

Proje Seçiminde Cevap Aranan Sorular?

- Hangi alanlara katkı sağlayacak
- Ne düzeyde katkı sağlayacak
- Ne kadar değerlidir
- Ne kadar anlamlıdır
- Hangi soruna/sorunlara nasıl çözüm bulmaktadır
- Özgün/yenilikçi yaklaşımı var mıdır?
- Kim/kimler için önemlidir. Hedef kitle!
- Geliştirilebilir yanları var mıdır? vb..

ÖRNEKLERLE PROJE İSİMLENDİRME
AMAÇ VE HEDEF BELİRLEME

Yazılım Mühendisliği

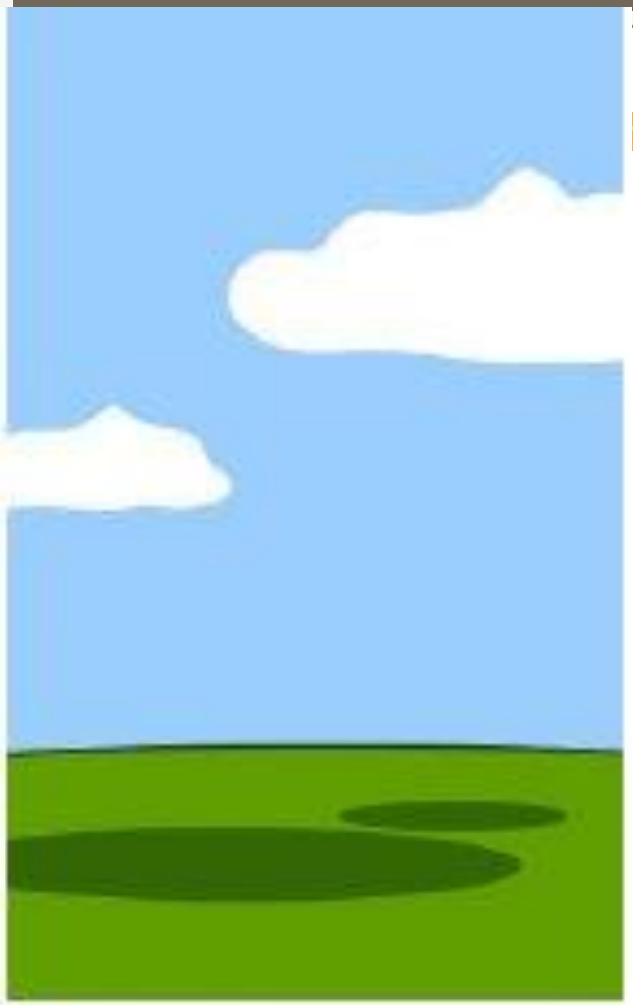
Bölüm - 3

Planlama

Müşterinin İstediği

Analistin Anladığı

Yazılımcının Geliştirdiği



Projenin Dokümantasyonu

Müşterinin Asıl İhtiyacı

Planlama

- Yazılım geliştirme sürecinin ilk aşaması
- Başarılı bir proje geliştirebilmek için projenin tüm resminin çıkarılması işlemi
- Proje planlama aşamasında yapılan işlemler
  - Proje Kaynaklarının Belirlenmesi
  - Proje Maliyetlerinin Kestirilmesi
  - Proje Ekip Yapısının Oluşturulması
  - Ayrıntılı Proje Planının Yapılması
  - Projenin İzlenmesi
- Proje planı tüm proje süresince sürekli olarak kullanılacak, güncellenecek ve gözden geçirilecek bir belgedir

Proje Kaynakları

- İnsan Kaynakları
- Donanım Kaynakları
- Yazılım Kaynakları

Planlama; bu kaynakların tanımını yapar ve

zaman kullanımı,

görev süreleri,

edinilme zamanlarını

planlar

İnsan Kaynakları

- Planlama; hangi tür elemanların, hangi süre ile ve projenin hangi aşamalarında yer alacağını belirler

Proje Yöneticisi	Donanım Ekip Lideri
Yazılım Ekip Lideri	Donanım Mühendisi
Web Tasarımcısı	Ağ Uzmanı
Sistem Tasarımcısı	Yazılım Destek Elemanı
Programcı	Donanım Destek Elemanı
Sistem Yöneticisi	Eğitmen
Veri Tabanı Yöneticisi	Denetleyici
Kalite Sağlama Yöneticisi	Çağrı Merkezi Elemanı

Donanım Kaynakları

- Günümüzde giderek açık sistem mimarisine dönüşmektedir.
- Donanım Kaynakları:
  - Ana Bilgisayarlar
  - Sunucular (Web, E-posta, Veri Tabanı)
  - Kullanıcı Bilgisayarları (PC)
  - Yerel Alan Ağı (LAN) Alt Yapısı
  - Geniş Alan Ağı (WAN) Alt Yapısı
- Yazılımın geliştirileceği ortam, gerçek kullanım ortamı dışında olmalıdır.
- Öte yandan, geliştirme ve uygulama ortamlarının aynı konfigürasyonda olmaları, ileride kurulum sırasında ortaya çıkabilecek taşıma sorunlarını büyük ölçüde giderecektir.

Yazılım Kaynakları

- Büyük ölçekte otomatik hale getirilmiş ve bilgisayar destekli olarak kullanılmaktadır.
- Bilgisayar Destekli Tasarım (CAD) ve Bilgisayar Destekli Mühendislik (CASE) araçları olarak bilinmektedirler.

Yazılım Kaynakları

- İş sistemleri planlama araçları
  - İş akış yapısının üst modelinin üretilmesinde kullanılır.
  - Bilgi akışı, bilgi yapısı iş birimlerindeki tıkanıklıklar bu araçlar kanalıyla ortaya çıkarılır.
- Proje yönetim araçları
  - Yönetici tarafından, projede yapılan işlerin izlenmesi, kaynak ataması, proje iş yapısının üretilmesi, gözlenen değerlerin işlenmesini sağlayan araçlar.
- Analiz ve tasarım araçları
  - Kullanılan modelleme tekniklerini ayrı ayrı ya da bütünleşik olarak uygulayan araçlar. Üretilen modelin kalitesinin ölçülmesi
- Programlama araçları
  - Derleyiciler, nesne-tabanlı programlama araçları, görsel programlama platformları.

Yazılım Kaynakları (devam)

- Test araçları
  - Yazılımı doğrulama ve geçerleme işlemlerinde kullanılır. Test verisi üreticiler, otomatik test yordamları, ...
- Prototipleme ve simülasyon araçları
  - Geliştirmenin erken aşamalarında kullanıcıya, sonuç ürünün çalışması ile ilgili fikir veren ve yönlendiren araçlar.
- Bakım araçları
  - Programın bakımını kolaylaştıran, bir kaynak koddan program şemalarının üretilmesini, veri yapısının ortaya çıkarılmasını sağlayan araçlar.
- Destek araçları
  - İşletim sistemleri, ağ yazılımları, e-posta ve ortam yönetim araçları.

Proje Maliyetleri

- Maliyet kestirimi; bir bilgi sistemi ya da yazılım için gerekebilecek iş gücü ve zaman maliyetlerinin üretimden önce belirlenebilmesi için yapılan işlemlerdir.

- Kullanılan Unsurlar
  - Geçmiş projelere ilişkin bilgiler
  - Proje ekibinin deneyimleri
  - İzlenen geliştirme modeli

  birden çok kez uygulanabilir

Proje Maliyetleri

Maliyet yönetimi sayesinde;

- Gecikmeler önlenir
- Bilgi sistemi geliştirme süreci kolaylaştırılır
- Daha etkin kaynak kullanımı sağlanır
- İş zaman planı etkin olarak gerçekleştirilir
- Ürün sağlıklı olarak fiyatlandırılır
- Ürün zamanında ve hedeflenen bütçe sınırları içerisinde bitirilir

Gözlemlenebilecek değerler

- Projenin toplam süresi
- Projenin toplam maliyeti
- Projede çalışan eleman sayısı, niteliği, çalışma süresi
- Toplam satır sayısı
- Bir satırın maliyeti (ortalama)
- Bir kişi/ay'da gerçekleştirilen satır sayısı
- Toplam işlev sayısı
- Bir işlevin maliyeti
- Bir kişi/ay'da gerçekleştirilen işlev sayısı
- Bir kişi/ay'da maliyeti

Proje Sınıfları

- Ayrık Projeler:
  - Boyutları küçük,
  - Deneyimli personel tarafından gerçekleştirilmiş
  - LAN üzerinde çalışan insan kaynakları yönetim sistemi gibi
- Yarı Gömülü:

  Hem bilgi boyutu hem donanım sürme boyutu olan projeler
- Gömülü Projeler:

  Donanım sürmeyi hedefleyen projeler (pilotsuz uçağı süren yazılım - donanım kısıtları yüksek)

İşlev Noktaları Yönetimi

İşlev Noktaları Yöntemi

İşlev noktaları, geliştirmenin erken aşamalarında (çözümleme aşamasında) saptanan bir değerdir. Bu değer, sistemin oluşturulduğu uygulama geliştirme ortamından bağımsız olarak elde edilmektedir. İşlev noktalarının hesaplamasında problem tanımı girdi olarak alınarak üç temel adım izlenir:

1. Problemin bilgi ortamının incelenmesi
2. Problemin teknik karmaşıklığının incelenmesi
3. İşlev noktası hesaplama

Özellikle, çözümleme çalışması, yapısal yöntemlerle yapıldığında, işlev noktaları yarı otomatik bir biçimde kolayca elde edilebilir. İşlev noktaları yönteminin proje geliştirme sürecinin erken aşamalarında oldukça etkili olduğu belirtilmektedir.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Girdileri

Yazılıma girdi olarak verilen her farklı uygulama bileşeni bir kulanıcı girdisi olarak sayılır. Kulanıcı girdilerinin sayısı hesaplanırken, alan bazında değil, daha genel olarak mantıksal kayıt bazında uygulama yapılmalıdır. Örneğin, bir personel sisteminin geliştirilmesinden söz edildiğinde, personel sicil bilgileri, personel izin bilgileri gibi bilgiler ayrı kullanıcı girdileri olarak sayılabilir. Personel sicil no, personel adı soyadı türündeki alan bilgileri kullanıcı girdisi olarak sayılmamalıdır.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Çıktıları

Kullanıcıyı ilgilendiren her tür mantıksal çıktı, kullanıcı çıktısı olarak sayılmalıdır. Bu kapsamda kullanıcı çıktılarına örnek olarak, raporlar, ekran çıktıları, hata iletileri vb. verilebilir. Girdiler gibi çıktılar da genel olarak sayılmalıdır. Bir raporun içerisindeki bir alan kullanıcı çıktısı olarak sayılmamalıdır. Öte yandan, aynı bilgileri içeren ancak bilgi düzeni ya da sıralaması farklı olan raporlar da ayrı kullanıcı çıktısı olarak ele alınmamalıdır. Örneğin, soyadına göre sıralanmış personel listesi ve sicil no'suna göre sıralanmış personel listesi raporları farklı kullanıcı çıktıları olarak sayılmamalıdır.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kullanıcı Sorguları

Kullanıcı sorgusu, çevrim içi olarak bilgisayara verilen bir girdi sonucu yine çevrim içi olarak bir kullanıcı çıktısı alınması biçiminde tanımlanmaktadır. Her farklı sorgu ayrı olarak sayılmalıdır. Örneğin; personel sicil bilgilerinin sorgulanması, personel maaş bilgilerinin sorgulanması vb.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Kütükler

Her mantıksal bilgi yığını ya da kütük ayrı olarak sayılmalıdır. Personel sicil kütüğü, personel maaş kütüğü, personel hareket kütüğü vb.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü		
		Yalın	Ortalama	Karmaşık
Kullanıcı Girdi Sayısı		3	4	6
Kullanıcı Çıktı Sayısı		4	5	7
Kullanıcı Sorgu Sayısı		3	4	6
Kütük Sayısı		7	10	15
Dışsal Arayüz Sayısı		5	7	10
Toplam Sayı				

Dışsal Arayüzler

Geliştirilmesi öngörülen bilgi sisteminin, gerek kurum içinde gerekse kurum dışında bir başka bilgi sistemi ile bilgisayar ortamında bir iletişimi (çevrim dışı disket vb. ya da çevrim-içi doğrudan bağlantı) söz konusu ise bu durum bir dışsal arayüz olarak sayılmalıdır. Kağıt ya da rapor türündeki arayüzler bir dışsal arayüz sayılmazlar. Örneğin, personel bilgi sistemi, yine aynı kurumda olduğu varsayılan muhasebe bilgi sistemine, personel maaşlarına ilişkin bilgiyi bilgisayar ortamında aktarması bir dışsal arayüz olarak sayılabilir.

- Bunların ağırlık faktörleriyle çarpımları toplanarak, Ayarlanmamış İşlev Nokta (AİN) sayısı hesaplanır.

Problem Teknik Karmaşıklığının İncelenmesi - TKF

- Bu adımda, probleme ilişkin dışsal etkenlerin değerlendirilmesi yapılır. Problem **Teknik Karmaşıklık Faktörü (TKF)**, tablo 3.3'de belirtilen ve problemin karmaşıklığına ilişkin sorulara verilecek yanıtların toplamıyla elde edilir. Uygulama ortamının ve uygulamanın özellikleri dikkate alınarak her soruya 0 ile 5 arasında bir yanıt verilir. Örneğin, 8. soruyu dikkate alalım. Eğer uygulama, ana veritabanının, tümüyle çevrim içi olarak güncellenmesini gerektiriyorsa bu durumda yanıtımız 5 olacaktır. Ancak, bazı veri tabanı işlemleri (örneğin %10 gibi) çalışma saatleri dışında, çevrim dışı olarak yapılıyorsa bu durumda 4 gibi bir yanıt verilebilir.

Yanıt Kılavuzu:

0: Hiçbir Etkisi Yok
1: Çok Az etkisi var
2: Etkisi Var
3: Ortalama Etkisi Var
4: Önemli Etkisi Var
5: Mutlaka Olmalı, Kaçınılamaz

1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?
2. Veri iletişimi gerekiyor mu?
3. Dağıtık işlem işlevleri var mı?
4. Performans kritik mi?
5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?
6. Sistem, çevrim içi veri girişi gerektiriyor mu?
7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?
8. Ana kütükler çevrim-içi olarak mı günleniyor?
9. Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı?
10. İçsel işlemler karmaşık mı?
11. Tasarlanacak kod, yeniden kullanılabilir mi olacak?
12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?
13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?
14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?

- Cevaplar 0 ile 5 arasında puanlandırılır
- Bunlar hesaplanıp toplanarak Teknik Karmaşıklık Faktörü (TKF) elde edilir.

İşlev Nokta Sayısı Hesaplama

Bu son adımda, bilinen ve sık kullanılan deneysel formül kanalıyla ilk iki adım sonucundaki bulgular kullanılarak işlev nokta sayısı belirlenir.

Gerek AİN, gerekse TKF önceki bölümlerde açıklandığı gibi hesaplandıktan sonra, geliştirilecek bilgi sistemine ilişkin İşlev Nokta Sayısı (İN) aşağıdaki formül kullanılarak hesaplanır.

İşlev Nokta Sayısı çeşitli amaçlarla kullanılabilir:
İşlev Noktaları, başka yöntemlerle birlikte kullanılmak istenildiğinde, kullanılacak yazılım geliştirme platformuna göre aşağıdaki değerler kullanılarak kolayca satır-sayısı kestirimine dönüştürülebilir.

İN = AİN x (0.65 x 0.01 x TKF)

Üretkenlik = İN / Kişi-Ay
Kalite = Hatalar / İN
Maliyet = $ / FP

İşlev Noktaları, başka yöntemlerle birlikte kullanılmak istenildiğinde, kullanılacak yazılım geliştirme platformuna göre aşağıdaki değerler kullanılarak kolayca satır-sayısı kestirimine dönüştürülebilir.

Örneğin İN kestirimi olarak 300 İN bulduğumuzu varsayalım. Eğer nesne-kökenli bir dil kullanarak (smalltalk gibi) yazılımı geliştirecek isek yaklaşık satır sayısı kestirimi **300x30=9000** satır olacaktır.

Programlama Platformu	Satır Sayısı/İN (Ortalama)
Assembly Dili	300
COBOL	100
FORTRAN	100
Pascal	90
C	90
Ada	70
Nesne-Kökenli Diller	30
4. Kuşak Dileri	20
Kod Üreticiler	15

Etkin Maliyet Modeli - COCOMO

Tüm COCOMO modelleri, temel girdi olarak satır sayısı kestirimini alır ve çıktı olarak iş-gücü (Kişi-ay, Kişi-yıl vb.) ve zaman (ay, yıl, hafta vb.) çıktılarını verir .

- İş Gücü (K) $K=a*S^b$
- Zaman (T) $T=c*K^d$

a,b,c,d : her bir model için farklı katsayılar

S : bin türünden satır sayısı

Proje Sınıfları

- **Ayrık Projeler:**
  - Boyutları küçük,
  - Deneyimli personel tarafından gerçekleştirilmiş
  - LAN üzerinde çalışan insan kaynakları yönetim sistemi gibi
- **Yarı Gömülü:**

  Hem bilgi boyutu hem donanım sürme boyutu olan projeler
- **Gömülü Projeler:**

  Donanım sürmeyi hedefleyen projeler (pilotsuz uçağı süren yazılım - donanım kısıtları yüksek)

Temel Model

- Küçük-orta boy projeler için hızlı kestirim yapmak amacıyla kullanılır
- Dezavantajı: Yazılım projesinin geliştirileceği ortam ve yazılımı geliştirecek ekibin özelliklerini dikkate almaz
- Avantajı: Hesap makinesi ile kolaylıkla uygulanabilir

Temel Model

- Ayrık Projeler
  - İş Gücü $K=2.4*S^{1,05}$
  - Zaman $T=2.5*K^{0,38}$
- Yarı Gömülü Projeler
  - İş Gücü $K=3,0*S^{1,12}$
  - Zaman $T=2.5*K^{0,35}$
- Gömülü Projeler
  - İş Gücü $K=3,6*S^{1,20}$
  - Zaman $T=2.5*K^{0,32}$

Araştırma ödevleri

- Projeniz için gereksinim analizini yazınız
- Süreç modelini belirleyiniz
- Maliyet kestirimini öğrendiğiniz yöntemlere göre hesaplayınız
- Gömülü ve yarı gömülü sistemlere örnekler veriniz

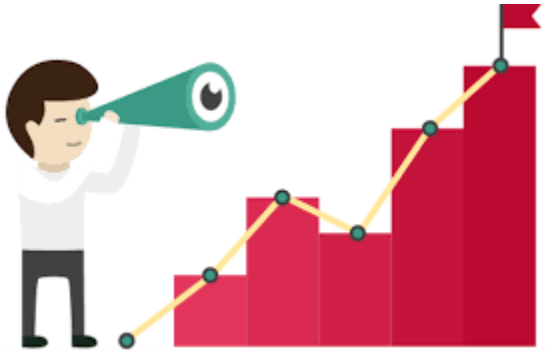
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

5.Hafta Sistem Çözümleme

Dr. Muhammet BAYKARA

Bölüm Hedefi

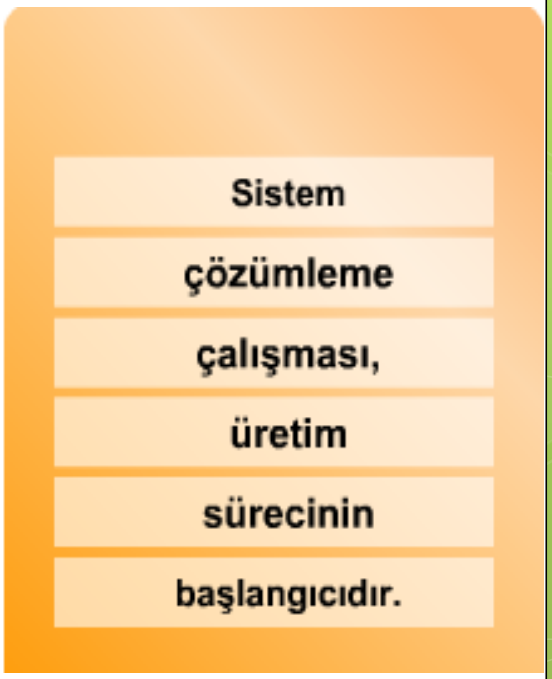
- Bu bölümde, yazılım geliştirme aşamalarından kullanıcı kesimini en çok ilgilendiren aşama olan çözümleme aşamasında kullanılan yöntem ve araçlar olan Kullanıcı arayüzü prototipleme, veri modelleme, süreç işlem modelleme ve veri toplama yöntemleri açıklanmaktadır.



Sistem Çözümlemeye Giriş

- Sistem çözümleme çalışması, üretim sürecinin başlangıcıdır. Bu aşamada temel olarak mevcut sistemin nasıl çalıştığı araştırılır.

- Mevcut sistemin incelenmesi sırasında temel hedef gereksinimlerin saptanmasıdır. Bu işlemden sonra önerilen sistem için modelleme yapılır. Söz konusu model, mantıksal model olarak bilinir.



Sistem Çözümleme

- Çözümleme çalışmasında mutlaka bir model/yöntem kullanma zorunluluğu vardır. Aksi durumda, çalışma dağınık biçimde sürer, denetlenemez ve başarısız olur.
- Bu bölümde, yapısal sistem geliştirme yaklaşımında kullanılan yöntemlerden en yaygın olanlarına örnekler verilmiştir. Yöntemler, veri modelleme ve süreç modelleme yöntemleri olarak iki başlık altında incelenmiştir.

Gereksinim Nedir?

- Bir sistem geliştirilirken, kullanıcının sistemin işlevleri ile ilgili beklentileri sistemin amaçlarını oluşturur. Gereksinim, sistemin amaçlarını yerine getirme yeteneği olan bir özellik ya da belirtim olarak tanımlanmaktadır.
- Bir kuruluş için personel bordro sistemi geliştirdiğimizi varsayalım. Maaş bildirim formlarının aylık olarak hazırlanması, kuruluşun değişik birimlerinden bu bilgilere erişim istekleri, gereksinimlere örnektir.

Gereksinim Türleri

İşlevsel Gereksinimler

Sistem ile çevresi arasındaki iletişimi belirleyen gereksinimler işlevsel gereksinim olarak tanımlanır. İşlevsel gereksinimler ayrıca sistemin herhangi bir durum karşısındaki davranışını belirler. Maaş çeki hazırlama örneğine dönecek olursak, işlevsel gereksinimler; maaş çekinin ne zaman hazırlanacağı, çek hazırlamak için ne tür girdiler verilmesi gerektiği, hangi koşullarda bir kişiye maaş verilmeyeceği gibi gereksinimler işlevsel gereksinimlerdir.

İşlevsel Olmayan Gereksinimler

Kullanıcının sorunundan bağımsız olarak çözülmesi gereken sorunlar işlevsel olmayan gereksinim olarak tanımlanır. Bir çok kaynakta bu terim yerine "Sistem Kısıtları" terimi de kullanılmaktadır. Kullanılacak bilgisayarın türü, yazılım geliştirme ortamı, kullanılacak veri tabanı yönetim sistemi vb. bu tür gereksinimlere örnek olarak verilebilir.

Tekrar

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Fiziksel Çevre

- İşlevlerin geliştirileceği, işletileceği aygıtlar nerededir?
- Sistem tek bir yerde mi olacak, birden çok, fiziksel olarak birbirinden ayrılmış yerler söz konusu mu?
- Sıcaklık, nem oranı veya manyetik etkileşim gibi çevresel kısıtlamalar var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Arayüzler

- Girdiler bir mi, yoksa birden çok sistemden mi geliyor?
- Çıktılar bir mi, yoksa birden çok sisteme mi gidiyor?
- Verinin nasıl biçimlendirileceğine ilişkin önerilen bir yol var mı?
- Verinin kullanılacağı, önerilen bir ortam var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Kullanıcı ve İnsan Etmeni

- Sistemi kim kullanacak?
- Çeşitli tipte kullanıcılar olacak mı?
- Her bir kullanıcı tipinin yetenek düzeyi nedir?
- Her kullanıcı tipi için ne tür bir eğitim gereklidir?
- Bir kullanıcının sistemi anlaması ne denli kolay olacaktır?
- Bir kullanıcının sistemi kötü amaclı kullanabilmesi ne ölçüde zordur?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

İşlevsellik

- Sistem ne yapacak?
- Sistem bunu ne zaman gerçekleştirecek?
- Sistem nasıl ve ne zaman değiştirilebilir veya güçlendirilebilir?
- Çalışma hızı, yanıt süresi ya da çıktı üzerinde kısıtlayıcı etmenler var mı?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Belgeleme

- Ne kadar belgeleme gereklidir?
- Belgeleme hangi kullanıcı kitlesini hedeflemektedir?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Veri

- Hem giriş hem çıkış için verinin biçimi ne olmalıdır?
- Bu veri ne sıklıkta alınacak veya gönderilecektir?
- Bu veri kesinlik (doğruluk) ölçüsü ne kadar olmalıdır?
- Hesaplamalar hangi duyarlık derecesine kadar yapılacaktır?
- Sistemden ne kadar veri akışı olacaktır?
- Belli bir zaman suresince veri kaynağında saklanacak mıdır?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Kaynaklar

- Sistemi kurmak, kullanmak ve bakımını yapmak için ne kadar malzeme, personel ve diğer kaynaklara ihtiyaç vardır?
- Geliştiriciler hangi yeteneklere sahip olmalıdır?
- Sistem ne kadar bir fiziksel yer kaplayacaktır?
- Güç (besleme), ısıtma veya soğutma için gereksinimler nelerdir?
- Geliştirim için tavsiye edilen bir zaman çizelgesi var mıdır?
- Geliştirme veya yazılım ve donanım icin harcanacak para üzerinde bir sınırlama var mıdır?

Gereksinimler
Fiziksel Çevre
Arayüzler
Kullanıcı ve İnsan Etmeni
İşlevsellik
Belgeleme
Veri
Kaynaklar
Güvenlik
Kalite Güvencesi

Güvenlik

- Sisteme veya bilgiye erişim denetlenmeli midir?
- Bir kullanıcının verisi diğerlerininkinden nasıl
- ayrıştırılacaktır?
  Kullanıcı programları diğer program ve işletim
- sistemlerinden nasıl ayrı tutulacaktır?
- Sistem hangi sıklıkta yedeklenecektir?
- Yedek kopyaları başka bir yerde mi saklanmaktadır?
  Yangın ve hırsızlığa karşı ne tür önlem alınacaktır?

Gereksinimler

- Fiziksel Çevre
- Arayüzler
- Kullanıcı ve İnsan Etmeni
- İşlevsellik
- Belgeleme
- Veri
- Kaynaklar
- Güvenlik
- Kalite Güvencesi

Kalite Güvencesi

- Güvenirlik için gereksinimler nelerdir?
- Sistemin özellikleri, insanlara nasıl aktarılmalıdır?
- Sistem, hataları kendisi bulup gidermeli midir?
- Sistem çökmeleri arasında öngörülen zaman nedir?
- Bir çöküm sonrasında sistemi tekrar başlatmak için uygun görülebilecek bir süre var mıdır?
- Sistem, tasarımda yapılacak değişiklikleri nasıl özümseyecektir?
- Bakımdan anlaşılan sadece hata gidermek midir, yoksa sistemi geliştirmeyi de içerir mi?
- Kaynak kullanımı ve yanıt süresine ilişkin verimlilik ölçütleri nedir?
- Sistemi bir yerden bir yere veya bilgisayardan diğerine taşımak ne kadar kolaylıkta olmalıdır?

Tekrar

Gereksinim Özellikleri

- Gereksinimler sadece bir sisteme ve sistemden bilgi akışını ve sistemce yapılan veri dönüşümünü tanımlamaz, aynı zamanda sistemin performansı üzerindeki kısıtları da tanımlar.
- Bu yüzden, gereksinimler üç amaca hizmet eder:
- Birincisi; geliştiricilerin, müşterilerin sistemin nasıl çalışmasını istediklerini anlamalarını sağlar.
- Gereksinimler, ayrıca tasarımcılara, sonuç sistemin ne işlevsellik ve özellikte olacağını söyler.
- Üçüncüsü; gereksinimler, sınama ekibine, kullanıcıyı, sunulan sistemin istenen sistem olduğuna ikna etmek için neler göstermeleri gerektiğini söyler.

Hem geliştiriciler hem de kullanıcılar gereksinimleri kullandığı için, gereksinimlerin geçerliliğinin doğrulanması gerekir. Doğrulama süreci aşağıda belirtilen yedi kriteri içerir:

1. **Gereksinimler doğru oluşturulmuş mudur?** Hem geliştirici, hemde kulanıcı tarafından belirsizliği giderilmiş, hatasız olarak onaylanmış olduğunu güvenceye almak için gözden geçiriniz.
2. **Gereksinimler tutarlı mı?** Yani, hiç çelişen, çift anlamlı, anlaşılmayan gereksinimler var mı?
3. **Gereksinimler tam mı?** Eğer tüm olası durumlar bir gereksinimle açıklanıyorsa, gereksinimler tamdırlar. Bu nedenle, örneğin bir çalışan ücretsiz izne çıktığında, birisi terfi ettiğinde veya bir kişi avans istediğinde, ne olacağı, örneğin bir PERSONEL sistemi tarafından açıklanmalıdır. Tanımlama, kullanıcı tarafından istenen tüm özellikleri içeriyorsa, bir sistem için "dışsal tam" dır denir. Bir tanımlama; eğer gereksinimler içersinde açıklanmamış referanslar yoksa "içsel tam" olarak tanımlanır.

Gereksinimlerin Geçerliliğinin Doğrulanması Kriterleri

4 **Gereksinimler gerçekçi mi?** Kullanıcının sistemden istediği gerçekten yapılabilir mi? Geliştirme zamanının uzun olduğu durumlarda, teknolojik değişime uyulabilecek biçimde tanımlı mı?

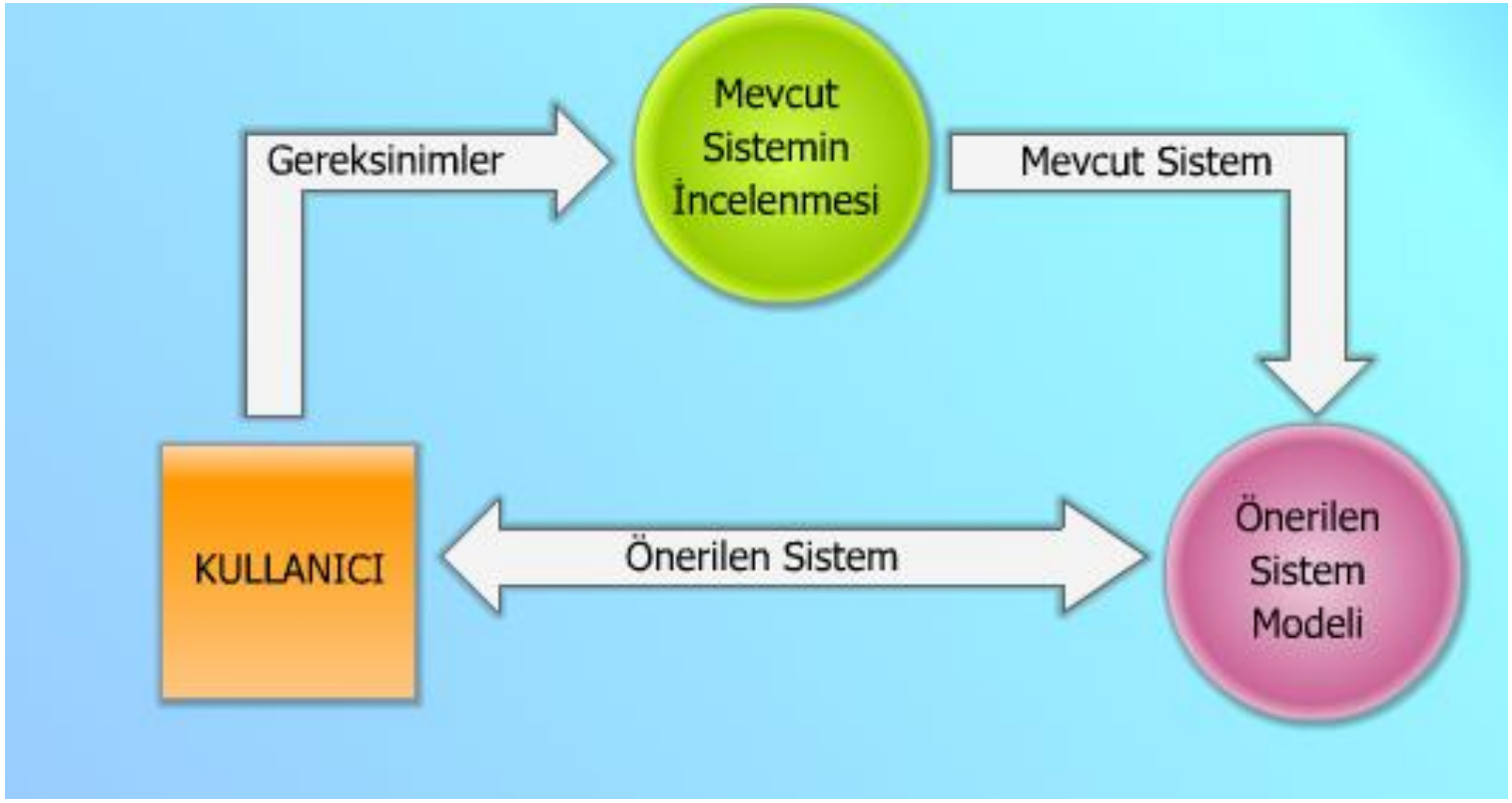
5 **Her gereksinim kullanıcı tarafından istenen bir şeyi mi tanımlamaktadır?** Bazı durumlarda bir gereksinim bizi gereksiz yere kısıtlayabilir. Örneğin, kullanıcı, ünü iyi olduğundan dolayı, belirli bir XYZ bilgisayarının kullanılmasını isteyebilir. Öte yandan, XYZ, istenen sistemin gerçekleştirmesi için en uygun bilgisayar olmayabilir. Bu tür gereksinimler doğrudan sistemin hedeflerini adreslemez. Yalnızca kullanıcının sorununu çözmek için doğrudan işe yarayacak gereksinimler üzerinde yoğunlaşılmalıdır.

Gereksinimlerin Geçerliliğinin Doğrulanması Kriterleri

6 **Gereksinimler doğrulanabilir mi?** Gereksinimlerin karşılandığını gösterebilmek için sınama senaryoları oluşturulabilir mi? Ya da doğrudan sınama yapılabilir mi?. Kullanıcı, bir hız tanımlaması yapmadan ve yanıt süresinin koşullarını tanımlamadan 'hızlı yanıt süresi' isteyebilir. Örneğin, 'hızlı yanıt süresi', daha doğrulanabilir olarak şu şekilde tekrar tanımlanabilir; 'sistemde kelime işlem programını kullanan en çok (örneğin: 32) sayıda kullanıcı ile, sistem bir kullanıcının ekranına bir belgeyi bir sonraki sayfasıyla birlikte en çok 5 saniyede getirmelidir.'

7 **Gereksinimler izlenebilir mi?** Her sistem işlevi, kendisini zorunlu kılan bir dizi gereksinim olarak izlenebilir mi? Sistemin özel bir yönüyle ilgili bir dizi gereksinim bulmak kolay mı? Örneğin, tüm iletişim gereksinimlerini gözden geçirmek için bütün gereksinimler okunmak zorunda mı?

Gereksinim Çözümleme Çalışması



Gereksinim Çözümleme Çalışması

- Geliştirilecek bilgi sistemi ya da yazılımla ilgili olarak tüm gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve bir gösterim biçimi ile açıklanması (modellenmesi) çalışmaları Sistem Çözümleme Çalışması olarak adlandırılır.

Temel olarak çözümleme çalışması:

- **Mevcut Sistemin İncelenmesi**
- **Önerilen Sistemin Modellemesi**

Olmak üzere iki aşamalı yapılır. Öncelikle mevcut sistem incelenir bu incelemenin tamamlanmasından sonra önerilen yapı modellenir.

Mevcut Sistemin İncelemesi

- Bu çalışmada temel amaç yazılım gelişitirilecek olan sistemin anlaşılması ve tanınmasıdır. Bu amaçla görüşme yapma, gerekirse anket yapma yöntemleri kullanılır.
- Yapılacak görüşmelerde, mevcut sistemde elle yürütülen tüm işlemler, girdi, işlev, çıktı ve diğer işlevlerle olan ilişkiler bazında sorgulanır. İlgili yönerge, kanun ve yöntemler kullanıcıdan edinilir.

Mevcut Sistemin İncelemesi

- Sorgulama ya da inceleme işleminin belirtilmesi amacıyla çeşitli yöntemler kullanılır. Sorgulama sonucu elde edilen bulgularla ilgili olarak kullanıcıdan geri bildirim alınır.
- Mevcut sistemde elle yürütülen işlerde, kullanılan form, defter ve yazışma örnekleri, sistemin veri boyutunu oluşturur. Bu tür bilgilerin birer kopyaları bir sonraki aşamada veri modellemesi için edinilir.

Önerilen Sistemin Modellemesi - Mantıksal Tasarım

- Mevcut sistemin modellenmesinden sonra, bilgisayarlı ortamda işlerin yapılabilmesi amacıyla önerilecek sistemin modeli oluşturulur. Bu model, önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünü içerir.
- Bu modelleme daha çok, bilgi sistemini geliştirecek teknik personele (sistem tasarımcıları, programcılar) yöneliktir. Bu model aynı zamanda '**mantıksal model**' olarak ta tanımlanır.

Önerilen Sistemin Modellemesi - Mantıksal Tasarım

- Mantıksal model, önerilen sistemin veri yapısını ve süreç yapısını hem genel hem de ayrıntılı olarak tanımlar. Mantıksal Model, kolaylıkla fiziksel modele (Program parçaları, veri tabanı tabloları vb) dönüştürülebilir bir yapıdadır.

- Bazı yazarlar, '**mantıksal model**' üretimini tasarım aşamasının bir parçası olarak ele alırlar ve konuyu '**tasarım**' başlığı altında incelerler.

Gereksinim Verisi Toplama

- Mevcut sistemin incelenmesi sırasında kullanılabilecek temel yöntemler:

  1. Sorma
  2. Psikolojik Türetme Teknikleri
  3. İstatistiksel Teknikler

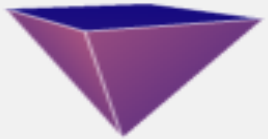
biçiminde özetlenebilir.

Sorma Yöntemi

- Sorma yöntemi, gereksinim verilerinin toplanması sırasında kullanılan en önemli yöntemlerden biridir.
- Karşılıklı görüşme ya da anket yolu ile uygulanır.

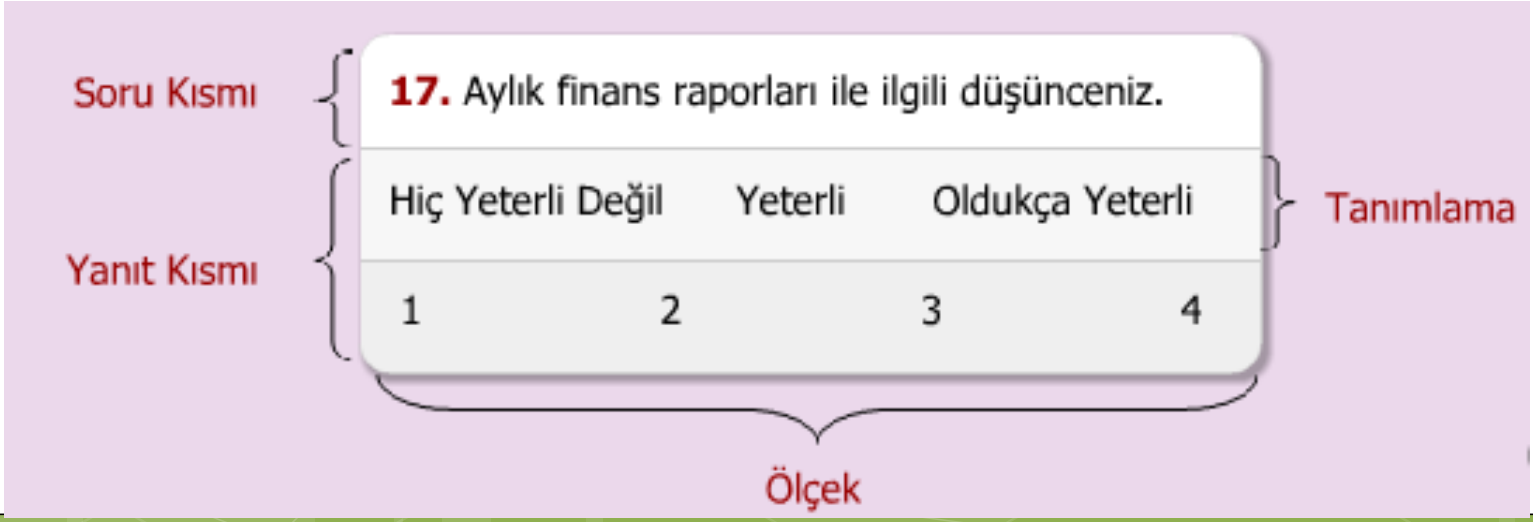
1. Karşılıklı Görüşme

- Karşılıklı görüşme sırasında, gereksinimleri ilişkin amaçlar, düşünceler, resmi olmayan yöntemler, duygular ve düşünceler araştırılır.
- Karşılıklı görüşme, veri toplama için en etkin yollardan biridir.

Piramit Tarzı	**Piramit Tarzı:** Özel sorularla başlayıp, giderek genel sorularla sürdürme.
 Koni Tarzı	**Koni Tarzı:** Genel sorularla başlayıp özel sorularla sürdürme.
Elmas Tarzı	**Elmas Tarzı:** Özel sorularla başlayıp, genel sorularla sürdürme, tekrar özel sorularla sona erdirme.

2. Anket

- Anket yöntemi, bir başka veri toplama yöntemidir. Kullanıcı sayısının fazla olduğu durumlarda, eğilimleri ve davranış biçimlerini saptamak amacıyla yapılır.
- Genelde yazılı test biçiminde hazırlanır. Bir anket sorusu temel olarak soru kısmı ve yanıt kısmı olmak üzere iki kısımdan oluşur.



Psikolojik Türetme Teknikleri

- Özellikle, belirsizliğin fazla olduğu ve zayıf yapılı ortamlarda, bilgi edinebilmek amacıyla insan psikolojisine dayalı teknikler kullanılır. Bu teknikler temelde görüşme ve ankete dayalı tekniklerdir. Diğerlerinden farkı, bilgi üretmek için psikolojide bilinen "üçleme" tekniğini kullanmasıdır.

- Bazı diğer psikolojik türetme yöntemleri, karar verme ortamlarında bilgi gereksinimlerini saptamak amacıyla algılama haritaları ve neden-etki çizgeleri kullanmaktadır.

İstatistiksel Teknikler

- Veri yoğun ve veri hacmi yüksek olan ortamlarda, verinin özelliklerini belirlemek amacıyla istatistiksel teknikler kullanılır.
- Bu yöntemlerden en çok bilinen ikisi **Örnekleme Yöntemi** ve **PIRA Modeli**dir.

İstatistiksel Teknikler

- **Örnekleme yöntemi**, bir topluluk içerisinden, sistematik yolla temsil edici örnek alma biçiminde tanımlanır. Amaç, veri toplama hızınını arttırmak ve verilerdeki çelişkileri önlemektir.
- Değişik örneklem türleri kullanılabilir; Basit gelişigüzel örneklem, kamaşık gelişigüzel örneklem, amaçlı örneklem vb. Örneklem boyutu, belirli bir güvenirlik düzeyinde olmak koşulu ile yine istatistiksel teknikler kullanılarak belirlenir.

İstatistiksel Teknikler

- **PIRA (Personal, Interactive, Report and Analysis) Modeli**, bilgi gereksinimlerinin tanımlarını belirli normlara bağlı olarak açıklamayı hedefler. Kişilerin bilgiye dayalı tercihlerini belirlemek amacıyla kulanılır.

Veri Modelleme Yöntemleri

- Önerilen sistemin mantıksal modelinde veri yapısını açıklamak amacıyla 'Veri Modelleme' yöntemleri kullanılmaktadır. Bu yöntemler, veri yapısını çeşitli düzeylerde tanımlama (en soyut düzeyden en ayrıntı düzeye kadar) amacını güder. Sistem Çözümleme aşamasında en yaygın olarak kullanılan veri modelleme yöntemleri:

- Nesne İlişki Şemaları
- Veri Sözlüğü

olarak bilinmektedir.

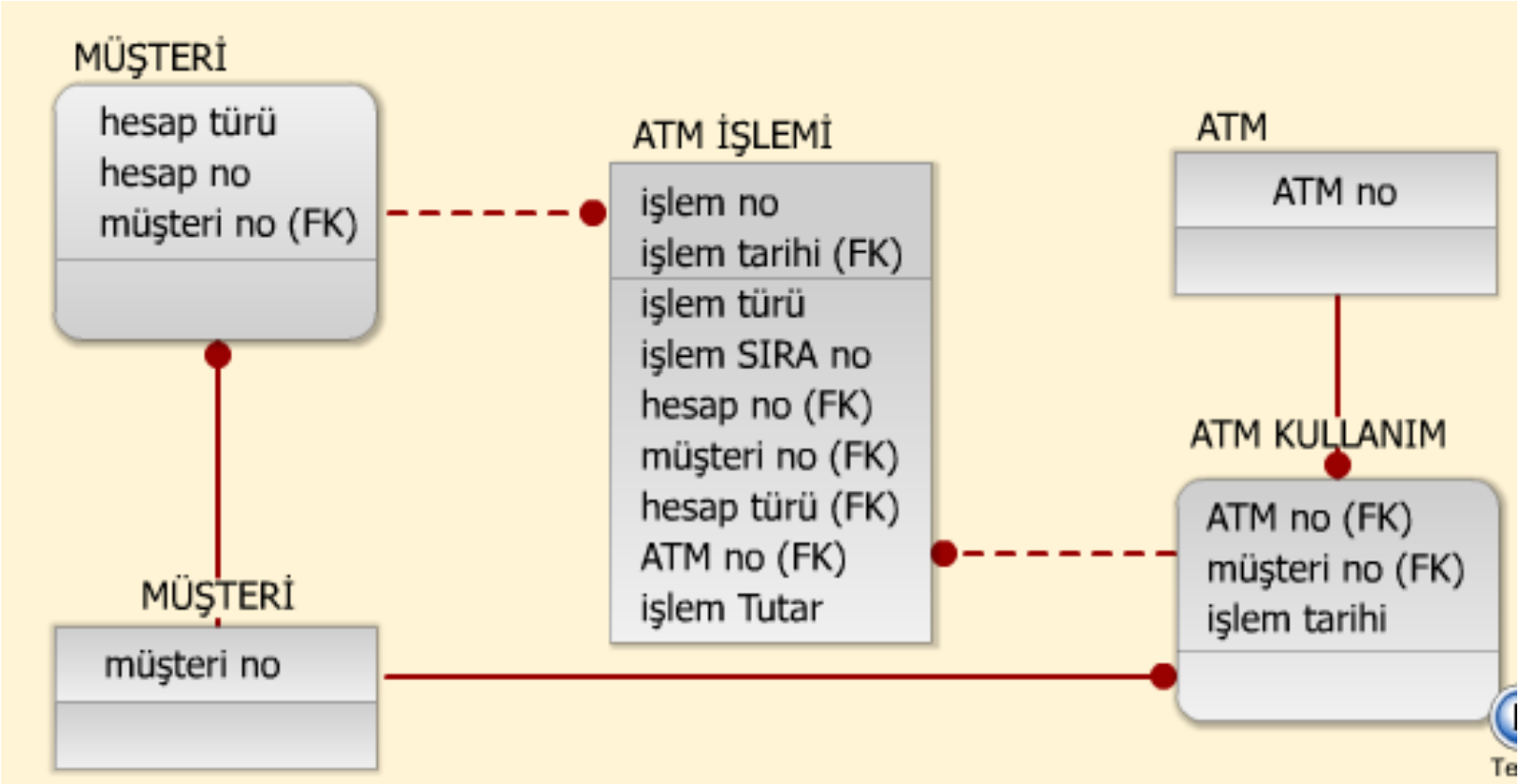
Nesne ilişki şemaları

- Bir veri nesnesi, üç temel özelliği ile bilinir:

  1. Veri nesnesi varlığının adı: Veri nesnesi varlığını tanımlayan özelliktir.
  2. Veri nesnesi varlığının özellikleri
  3. Veri nesnesi varlığının diğer veri nesnesi varlıklarına referansı: Veri nesnesi varlığının diğer veri nesneleri ile olan ilişkisinin belirtilmesi amacıyla kullanılır. Bu amaçla her bir veri nesnesini tek olarak belirleyen bir belirteç (anahtar) kullanılır. Söz konusu belirteç veri nesnesi varlığının ad özellikleri arasında yer alır.

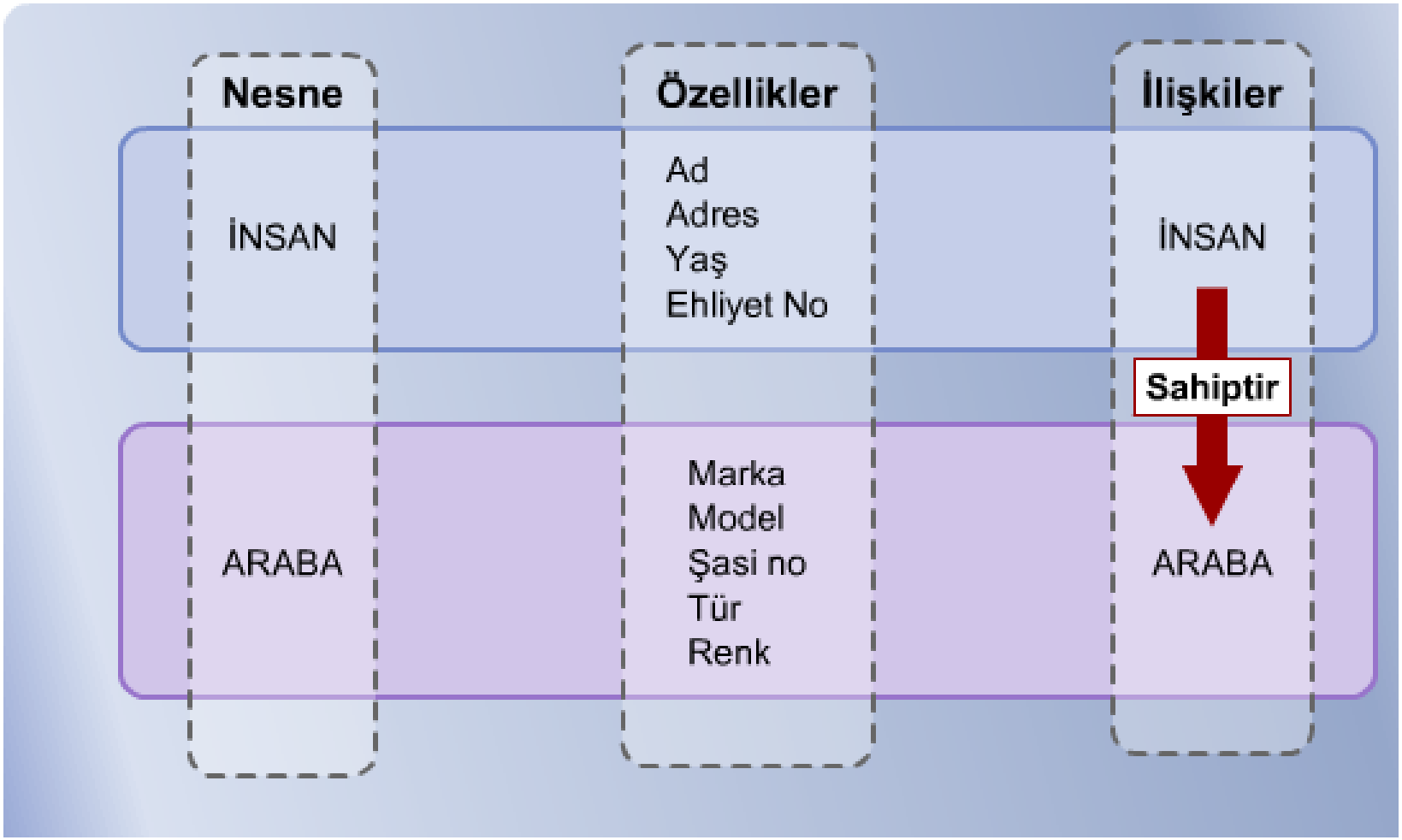
Nesne-İlişki Şemaları

- Geliştirilecek sistemin kullanacağı ana veri nesneleri ve aralarındaki ilişkileri belirtir.



Örnek - 1

ARABA ve **İNSAN** adlı iki veri nesnesi düşünelim.

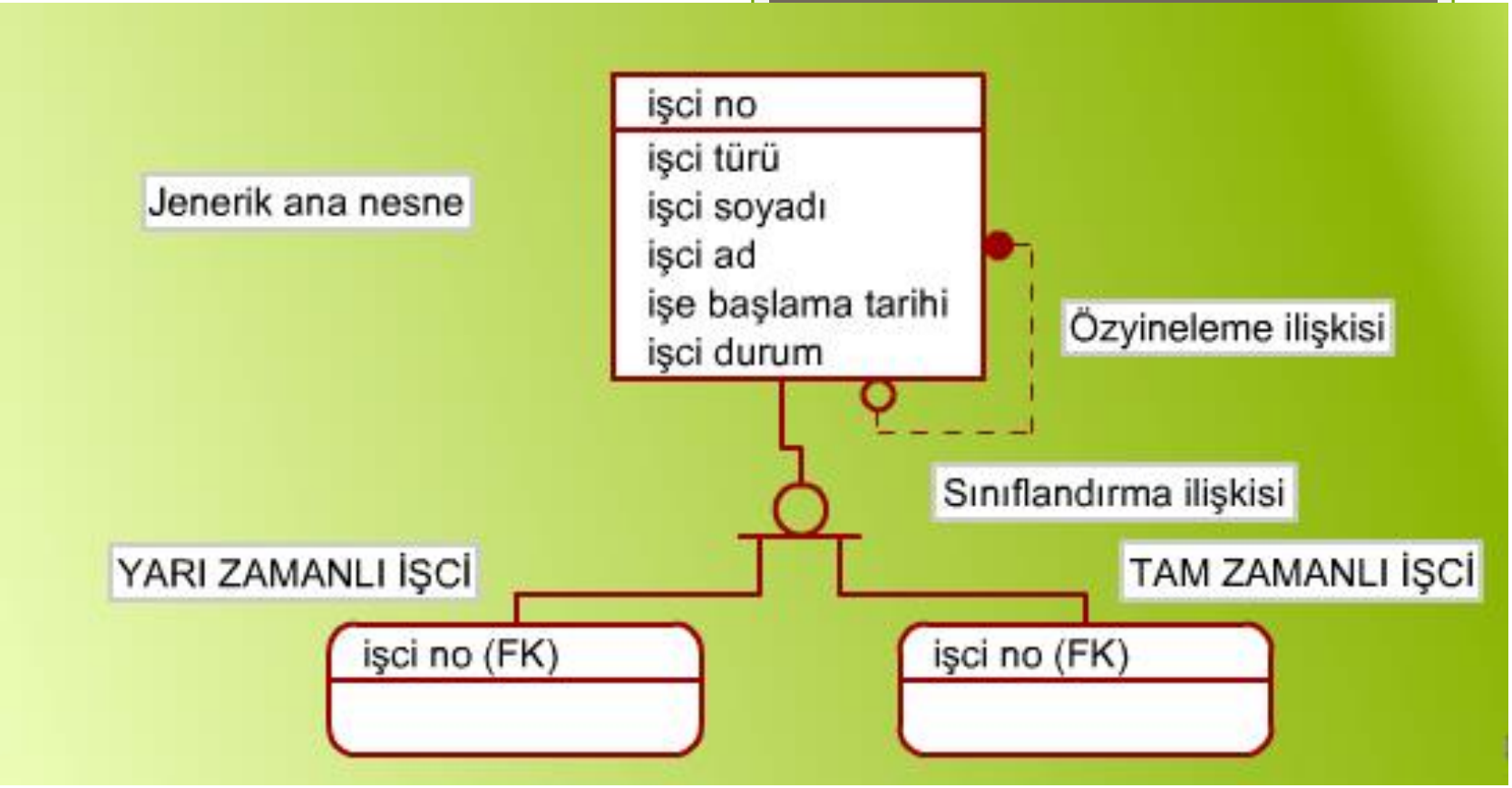


Yukarıda **ARABA** ve **İNSAN** nesnelerinin özellikleri verilmektedir. Araba nesnesinin varlıklarına ilişkin örnekler is

Marka	Model	Şasi no	Tür	Renk	Sahibi
BMW	740L	XZ68…	Sedan	Siyah	AA
VOLVO	S80	A32XT..	Sport	Sarı	BT
Ford	Taurus	X24R56	Hatchback	Bej	NB

Anahtar Alan

Te



Veri nesneleri arasındaki ilişkiler bire bir (1-1), bir den çoğa (1-N ya da N-1) ya da çoktan çoğa (M-N) tanımlanabilir. Örneğin:

1 - 1 ilişki : Bir İNSAN ancak bir ARABA sahibi olabilir.

1 - N ilişki: Bir İNSAN birden çok ARABA sahibi olabilir:

M - N ilişki: Birden çok İNSAN birden çok ARABA sahibi olabilir.

Veri Sözlüğü

- Nesne İlişki şemalarında belirtilen nesne özelliklerinin ayrıntılı tanımları Veri Sözlüğünde yer alır.

- Söz konusu ayrıntılı tanımlar genel olarak:
  - Veri Adı
  - Veri Eş-adı (Aynı veri için kullanılan diğer ad)
  - Nerede/nasıl kullanıldığı
  - İçerik tanımı

türünde bilgileri içerir.

Örnek: Kişi telefon bilgisinin tanımlanması

telefon no = [yer kodu | numara]
yer kodu = [212 | 242 | ..... | 312]
numara = * yedi basamaklı herhangi bir sayı *

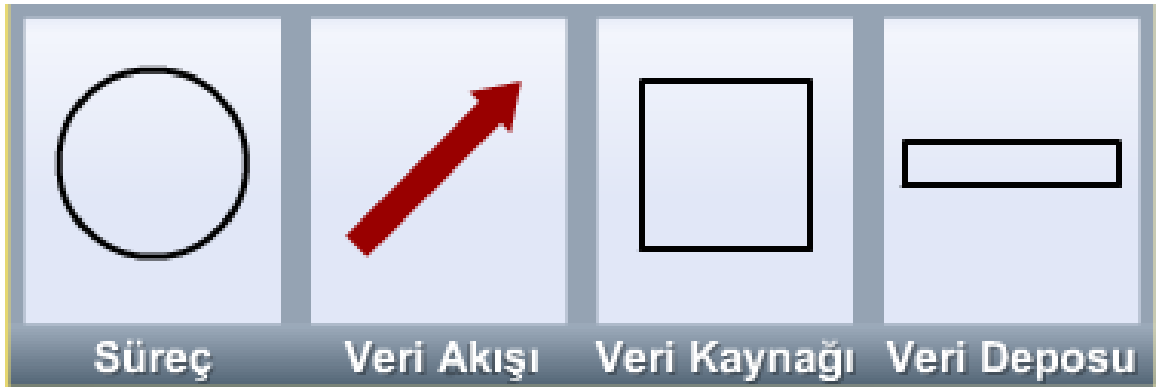
Veri Yapısı	Gösterim	Açıklama
	=	.. den oluşmuştur
Sıra Tanımlama	+	ve
Seçim	[|]	ya da
Yineleme	{ }^{n}	n kez yinelenir
	( )	seçenek veri
	* *	açıklama

Süreç/İşlem Modelleme Yöntemleri

- Süreç/İşlem modelleme yöntemleri, geliştirilecek sistemin süreç ya da işlemlerini ve bu süreçler arasındaki ilişkileri tanımlamak amacıyla kullanılan yöntemlerdir. En yaygın olarak kullanılan Süreç Modelleme Yöntemlerine örnek olarak:

Veri Akış Diyagramları

- VAD kullanılarak, geliştirilecek sistemin mantıksal modeli, 'Yukarıdan Aşağıya' bir yaklaşımla oluşturulur.
- Sistem önce en genel biçimiyle ele alınır, yalnızca dışsal ilişkileri incelenir. Daha sonra, sistemin iç yapısındaki süreçler ve bu süreçler arasındaki ilişkiler belirlenen bir ayrıntı düzeyine kadar modellenir.



VAD kümesi

- Temel olarak bir sistemin mantıksal modelinin süreç yapısı, üç tür Veri Akış Diyagramı çizilerek elde edilir.

Kapsam Diyagramı

Geliştirilecek bilgi sisteminin dışsal ilişkilerini göstermek amacıyla kulanılır. Yalnızca bir çember ve gerekli sayıda veri kaynağı ile aralarındaki veri akışlarının belirtiminden oluşur.

Genel Bakış Diyagramı

Bir Bilgi sisteminin ana işlevlerini, bu işlevlere ilişkin veri kaynaklarını, veri depolarını ve işlemler arasındaki ilişkileri içeren VAD, Genel Bakış Diyagramı olarak tanımlanmaktadır. Genel Bakış Diyagramı, kapsam diyagramının ayrıştırılmış ya da "patlatılmış" biçimidir. Bir bilgi sistemi için, bilgi sisteminin büyüklüğüne göre en çok bir kaç Genel Bakış Diyagramı çizilir.

Ayrıntı Diyagramı

Genel Bakış Diyagramında belirtilen her ana işlev, belirli bir ayrıntı düzeyine kadar (en fazla 7+/- 2) detay VAD oluşturularak ayrıştırılır. Genellikle detay VAD'larda, Veri kaynakları gösterilmez.

Tek

VAD Neyi Gösterir?

1. VAD, bilgi sisteminin durağan yapısını gösterir.

2. VAD, bilgi sisteminin süreçlerini, bu süreçler arasındaki veri akış ilişkilerini gösterir.

3. VAD, bilgi sistemi süreçleri ile ilgili olan kurum birimlerini ya da dış birimleri bilgi kaynakları olarak gösterir.

4. VAD, bilgi sistemi için gerekli olan ana veri depolarının neler olduğunu ve hangi süreçler tarafından kullanıldığını gösterir.

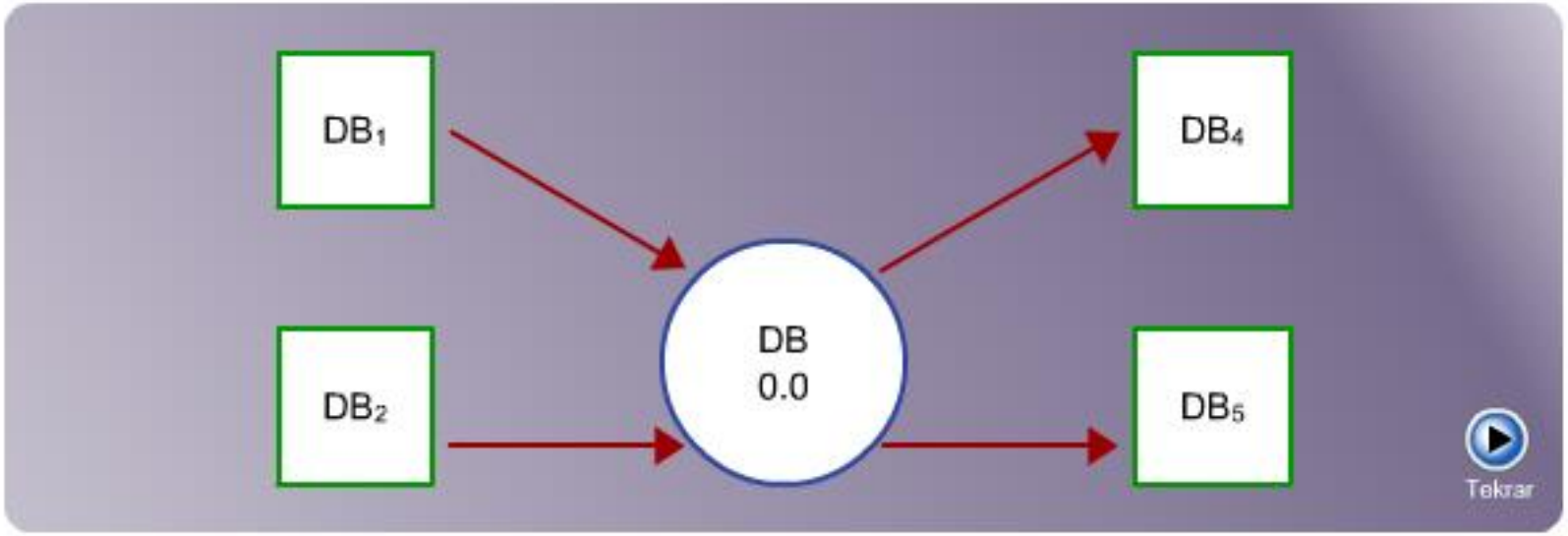
5. VAD, bilgi sistemi süreçlerini, yukarıdan aşağıya doğru ayrıştırarak gösterir. Böylelikle süreçler ve aralarındaki ilişki, en soyut (genel) düzeyden en ayrıntılı düzeye kadar belirli bir sıra düzen içerisinde belirtir.

VAD Neyi Göstemez?

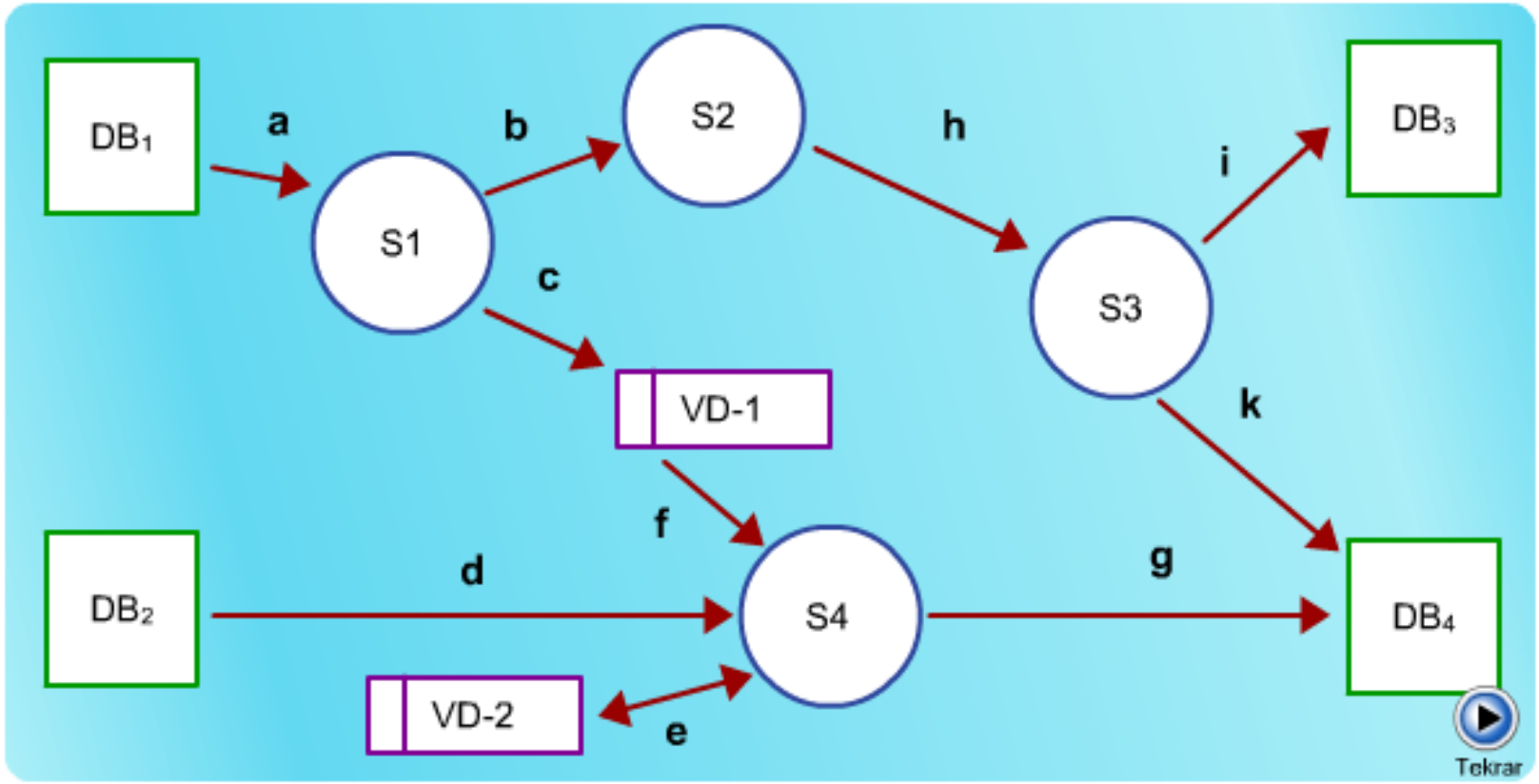
1. VAD bilgi sistemi süreçlerinin zamana ilişkin durumunu ve bu durumla ilgili bilgileri göstermez.

2. VAD, bilgi sistemi süreçlerinin kendi aralarındaki "karar" ilişkisini göstermez.

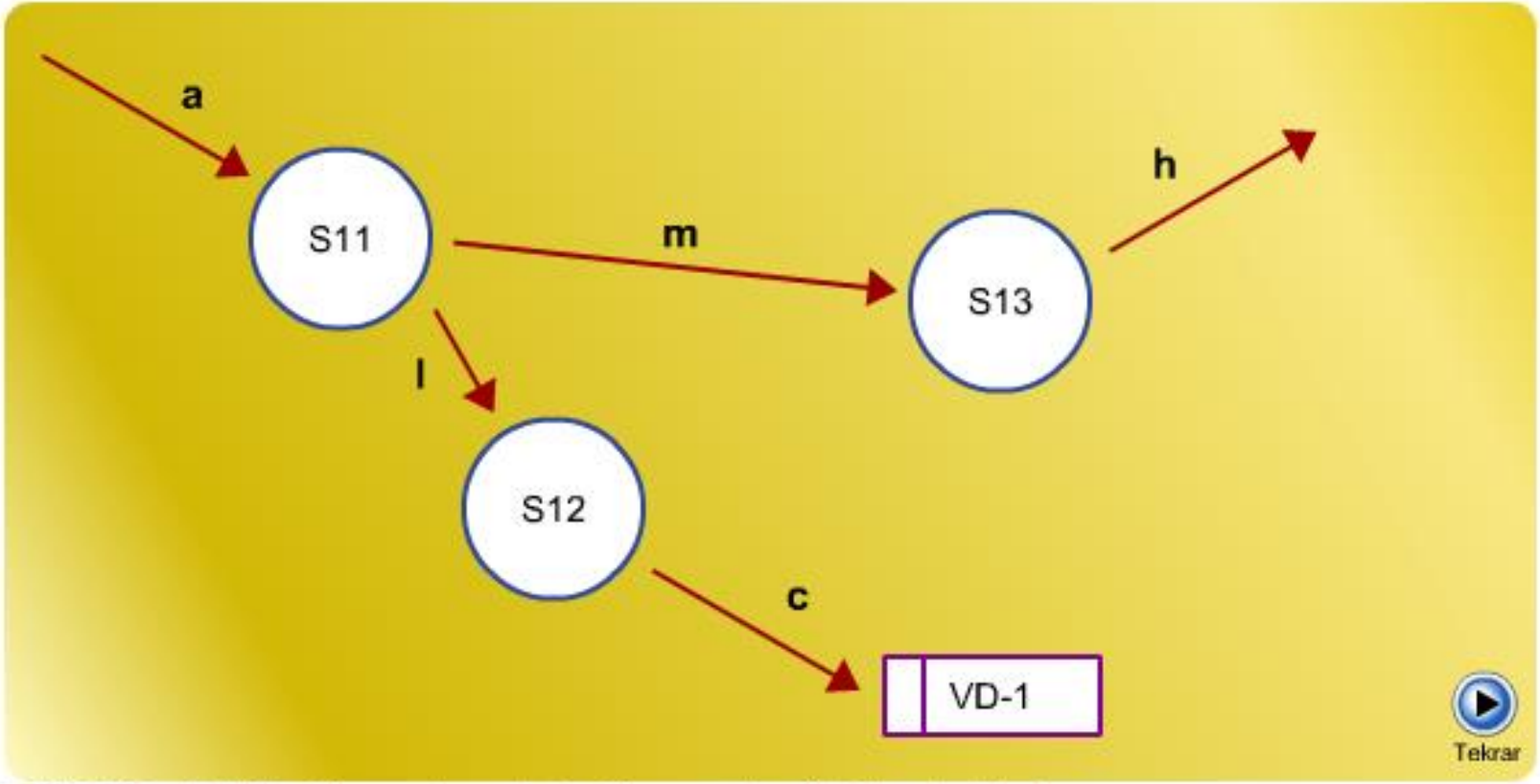
3. VAD, gerek bilgi sistemi süreçleri, gerek veri akışları gerekse bilgi kaynakları ve bilgi depoları için ayrıntı içermez.



Şekil 4.10a: Veri Akış Diyagramları - Kapsam Diyagramı Örneği
(DB: Dış birim, BS: Bilgi Sistemi)



Şekil 4.10b: Veri Akış Diyagramları - Genel Bakış Diyagramı Örneği
(S: Süreç, VD: Veri Deposu, a,b,c,d,e,f,g,h,i,k : Veri akışları)



Şekil 4.10c: Veri Akış Diyagramları - Detay Diyagram Örneği (Süreç-1 detayı)
(a,b,c,l,m : Veri akışları)

Süreç Tanımlama Dili (STD)

- Veri akış diyagramlarında isimleri belirtilen, aralarındaki ilişkiler gösterilen ve yukarıdan aşağıya ayrıştırılmış olan bilgi sistemi süreçlerinin iç yapılarını belirtmek amacıyla kullanılan araç, yöntem ya da gösterim biçimleri Süreç tanımlama dili olarak tanımlanmaktadır.

- Düz Metin,
- Şablon,
- Yapısal İngilizce

Düz Metin

Üçgeni İncele süreci üçgenin kenar boyutlarını (A,B,C) girdi olarak alır. Süreç önce bütün bu değerlerin pozitif olup olmadığını denetler. Eğer değerlerden biri negatif ise hata iletisi verir. Süreç, tüm kenar boyutlarının bir üçgeni belirleyecek biçimde geçerli olup olmadığını denetler. Eğer geçerli değerler ise üçgenin türünü (eşkenar üçgen, ikizkenar üçgen, çeşitkenar üçgen) belirler. Değil ise hata iletisi verir.

Şablon

Süreç: Üçgeni İncele
Girdi: Üçgenin kenar boyutları (A,B,C)
Çıktı: Üçgen Türü, Hata iletisi
İşlem: A,B,C değerlerin pozitif olup olmadığını denetle. Eğer değerlerden biri negatif ise hata iletisi ver. A,B,C değerlerinin geçerli olup olmadığını denetle. Eğer geçerli değerler ise üçgenin türünü (eşkenar ücgen, ikizkenar üçgen, çesitkenar üçgen) belirle. Değil ise hata iletisi ver.

Yapısal İngilizce

Procedure: Üçgeni İncele

Üçgenin kenar boyutlarını oku;
if herhangibir boyut negatif **then** hata iletisi ver; **endif;**
if en büyük kenar diğer ikisinin toplamından küçük **then**
  begin
  eşit kenar sayısını belirle;
  if üç kenar eşit then tür eşkenardır; **endif;**
  if iki kenar eşit then tür ikizkenardır; **endif;**
  if hiç bir kenar eşit değil then tür çesitkenardır; **endif;**
  üçgen türünü yaz;
end;
else değerler üçgeni belirtmiyor iletisi ver; **endif;**
endproc

Tekrar

KULLANICI ARAYÜZ PROTOTİPLEME (KAP)

- Gereksinim tanımlama ya da sistem çözümleme çalışmasının önemli bir bileşeni, oluşturulacak bilgi sistemine ilişkin girdi ve çıktı gereksinimlerinin tanımlanmasıdır.
- KAP Yöntemi, gereksinim çalışmasının hemen sonunda kullanıcıya gösterilecek bir prototip yazılım hazırlanmasını içermektedir. Söz konusu prototipin gerçekten içsel olarak çalışmayan ancak ekranlar, menüler ve bunların aralarındaki geçişlerin çalıştığı bir yazılımdır.

KAP İçin Ekranlar Nasıl Hazırlanır?

K

1. KAP hazırlama süresi uygulamanın boyutu ne olursa olsun, ekranlar ve ekranlar arası geçişlerin hazırlanması için gerekecek iş gücü ve zaman, sistem çözümleme için ayrılan zamanın %5'ini aşmamalıdır.

A

2. Bir özellik yalnızca bir kez gösterilmelidir. Bir özellik birden fazla ekranda yer alsa bile yalnızca bir ekranda gösterilmelidir. Örneğin, şehir listesini içeren bir kombo listesi, bir kaç şehir bilgisi içermeli ve yalnızca bir ekranda kulanıcıya gösterilmelidir. Başka ekranlarda aynı liste varsa, gösterilen ile aynı olduğu belirtilmelidir.

P

3. KAP hiçbir içsel işlem içermemelidir. Örneğin: bir ekranda kişilerle ilgili sorgulama yapıldığını düşünelim. Adı soyadı bilinen bir kişinin, bu bilgi verildiğinde, diğer ayrıntı bilgilerinin ekranda görüntüleneceğini varsayalım. Kullanıcı, Ad Soyad bilgisini girecek ve ilgili sorgulama tuşunu etkinleştirdikten sonra ekranda hep aynı bilgiler görünecektir. Yani, KAP, Ad soyad bilgisine göre KİŞİ tablosundan arama yapmayacak, program içerisinde hep aynı sonucları ekrana verecektir. Çünkü Kişi tablosunun yaratılması ya da kullanılması söz konusu değildir. Burada amaç, kullanıcıya, sorgulama biçimini ve sorgulama sonucunda elde edilecek bilgileri açıklamaktır. Sorgulanan bilginin içeriği bu aşamada önemli değildir. Kulanıcıdan, sorgulama sonucu gelen bilgi türlerine ilişkin, sorgulama işleminin işlem akışına ilişkin bilgi alınması hedeflenir.

KAP İçin Raporlar Nasıl Hazırlanır?

- Bilgi sisteminden yazıcı çıktısı biçiminde alınması istenen raporlar, bir metin düzenleyici (örneğin MS Word) aracılığı ile hazırlanır ve belirli bir biçimde numaralandırılır.

ÖRNEK

Geliştirilecek bilgi sisteminin İnsan Kaynakları Yönetim Bilgi Sistemi (IKYS) olduğunu varsayalım. IKYS Atama Modülüne ilişkin raporların örneklerini MS Word ile hazırlayarak bu raporlara IKYS-AT-R1, IKYS-AT-R2... vb. numaralar verelim. Kullanıcı için hazırladığımız prototipi, kullanıcıya gösterirken, kullanıcı rapor almak amacıyla ilgili tuşa bastığında ya da fare imleci ile ilgili rapor alma düğmesini etkinleştirildiğinde KAP "Şu anda yazıcıdan IKYS-AT-R1 no'lu rapor dökülmekte" iletisi türünde bir ileti verecektir. Kullanıcı bu iletiyi gördüğünde, daha önceden MS Word ile hazırlanmış olunan ve IKYS-AT-R1 biçiminde numaralanmış raporu kullanıcı ya göstererek, ileride yazılımı bitirip, sistemi hazır hale getirdiğimizde bu rapor un aynısının yazıcıdan alınacağını belirtiriz. Kulanıcı bu durumda bize, rapor alanları, raporun tetiklendiği ekranla ilgili geri bildirimlerini ve değişiklik isteğini iletir. Bu değisiklikler yapıldıktan sonra söz konusu raporla ilgili gereksinimler kesinleştirilmiş olur.

ÇÖZÜMLEME ÇALIŞMASI NASIL DEĞERLENDİRİLİR?

- Sistem çözümleme çalışması sonuçlandıktan sonra, elde edilen ara ürünün (mantıksal model) istenenleri karşılayıp karşılamadığının belirlenmesi amacıyla değerlendirilmesi gerekir. Bu nedenle temel olarak:

Kullanıcıların benimsemesi ve onayı

Tutarlılık, tamlık ve uygunluk

Tasarım çalışması için yeterliliği

Tamlık ve Tutarlılık

Tamlık, bilgi sitemi ya da yazılımın tüm öğeleri ve bunların arasındaki ilişkilerin tanımlanmasını gerektirir. Tutarlılık ise, önerilen modelin kendi içerisinde hatasız, çelişkisiz olması anlamındadır. Tamlık ve tutarlılık denetimi, bir programlama dilinde yazılmış bir programın söz dizim kurallarının denetimine benzer. Örneğin, modelleme aracı olarak VAD kullanıldığında;

- Bütün süreçlerin girdi ve çıktıları var mı?
- Süreçler, veri akışları, dış birim ve veri depoları uygun şekilde adlandırılıp numaralandırılmış mı?
- VAD' de ki tüm isimler veri sözlüğünde yer almış mı?
- Planlama aşamasında öngörülen belirtimlerin hepsi ele alınmış mı?

Olurluluk

- Olurluluk, sistem çözümleme sırasında yapılan çalışmanın, planlama aşamasında yapılan çalışmaya uygunluğunun belirlenmesi için yapılan çalışmaları içerir.
- Maliyet kestirim çalışması yinelenir, sapmalar saptanır, kaynaklar yeniden planlanır. Sapmalar olduça fazla olursa, yapılan çalışmanın yeniden gözden geçirilmesi gerekir.
- Bu aşamada, maliyet kestirimi için daha fazla bilgi mevcuttur. VAD yöntemi, maliyet kestirim yöntemlerinin kolayca uygulanabilmesini olanaklı kılar. Örneğin İşlev Noktalarının belirlenmesinde VAD'den:

VAD'den elde edilenler

Girdi ve Çıktı Sayıları: Girdi ve Çıktı sayıları doğrudan genel bakış diyagramındaki veri akışlarının incelenmesiyle elde edilir.

Kütük Sayısı: Genel bakış diyagramlarındaki veri depoları sayılarak elde edilir.

Sorgu Sayısı: Sorgu sayısı genel bakış diyagramı ve detay diyagramlardaki süreçlerin incelenmesiyle elde edilir.

Arayüz Sayısı: Arayüz sayısı, kapsam diyagramındaki dış birimlerle olan ilişkiler incelenerek elde edilir.

Alınan Dersler

- Sistem Çözümleme aşaması, uç kullanıcı ile iletişimin en fazla olduğu aşamadır. Kullanıcıların Bilişim teknolojileri konusundaki bilgi düzeyleri genelde yok denecek ölçüde düşüktür. Bu nedenle kullanıcı ile olan iletişimin olabildiğince kullanıcıya görsel olanaklar sunularak yapılması önemlidir. Bu nedenle, özellikle kullanıcı arayüzünün belirlenmesi için en etkili yöntem olarak Kullanıcı Arayüz Prototipleme yöntemi önerilmektedir.

- KAP yöntemini kullanıcılarla birlikte tartışırken, "iş senaryoları" oluşturmanın oldukça yararı vardır. Her iş senaryosu, bir iş probleminin çözümüne karşılık gelecek biçimde hazırlanmalı ve KAP üzerinde sınanmalıdır. BU yolla, kullanıcı, işeride bilgisayarlı uygulamaya geçildiğinde, nasıl bir ortamla karşı karşıya geleceğine ilişkin fikir sahibi olur.

Bu yöntemin kullanılması ileride ortaya çıkabilecek belirsizlik ve riskleri de büyük ölçüde ortadan kaldırır. Üretim sonucunda elde edilecek ekran ve rapor görüntülerinin bu aşamada olabildiğince kesinleştirilmesi, hem üretim yapan yazılım mühendislerinin hem de kullanıcının işini oldukça kolaylaştırır.

Alınan Dersler

Kullanıcılar, genellikle görüşme tutanaklarını imzalamakta çekingenlik gösterirler.'Siz hazırlayın, bize gönderin bizde imzalar size göndeririz ' biçiminde yöntemler önerirler.

Görüşme tutanaklarınızı anında iki kopya olarak tutun ve hemen görüşme bitiminde imzalatma yöntemini benimseyin ve uygulayın. Aksi durumda, ileride ortaya çıkabilecek sorunlarda kaybeden taraf siz olursunuz.

Alınan Dersler

Tüm görüşme ve toplantı kayıtlarınızı olabildiğince bilgisayarlı ortamda saklayın. Aradan, uzun bir süre geçtikten sonra, eski kayıtlara erişimde büyük kolaylıklar sağladığını göreceksiniz.

Bu amaçla Visual Source Safe türü basit bir araç bile kullanabilirsiniz.

Alınan Dersler

VAD yöntemi, kullanıcı tarafından algılanma açısından oldukça etkili bir yöntemdir. Bu durum, değişik projelerde görülmüştür.

Kullanıcıya, görüşme ile ilgili olarak düzeylendirilmemiş bir VAD hazırlayıp gönderdiğinizde, kullanıcının bu VAD'nı kendi başına yeniden çizebildiği gözlemlenmiştir.

Alınan Dersler

CASE aracı kullanımının modelleme çalışmasını oldukça hızalandırdığı gözlemlenmiştir. Niteliği ne olursa olsun mutlaka bir CASE aracı kullanın.

CASE aracı kullanımı ayrıca, daha sonraki aşamalarda oluşabilecek ve mantıksal modele yansıtılması gereken günlemelerin kolayca yapılmasını sağlar.

Sorular

1. Sistem Çözümleme çalışmasının amaç ve önemini belirtiniz.

2. Gereksinim modelleme çalışmasında neden grafiksel araç ve yöntemler daha sıklıkla kullanılır?
3. Mevcut sistemin incelenmesi için kullanılabilecek yöntemleri açıklayınız.
4. İş senaryosunu tanımlayınız. Bir PERSONEL bilgi sistemi uygulaması için üç iş senaryosu örneği veriniz.
5. Çevrenizde var olan bir CASE aracını inceleyip, sistem çözümleme çalışması ile ilgili olarak hangi olanakları içerdiğini araştırınız.
6. Gereksinimlerin belirlenmesi sırasında ortaya çıkabilecek riskleri sıralayınız.

Sorular

7. Beş adet açık uçlu soru örneği veriniz.
8. Beş adet kapalı uçlu soru örneği veriniz.
9. İstediğiniz bir konuda 20 soruyu içeren bir anket hazırlayınız. Anket yanıtlama problemlerini dikkate alınız.
10. Kaset, CD satan ve kiralayan bir müzik dükkanında yapılan işlemleri, yapısal VAD kullanarak modelleyiniz. Süreç tanımlama dili olarak düz metin kullanınız. Veri yapısını Nesne-İlişki diyagramları ile modelleyiniz.

11. 8. Soruda oluşturduğunuz VAD üzerinde

- Dönüştürme akış özelliklerini,
- Ara işlem akış özelliklerini

gösteriniz.

Sorular

12. Kullanıcı arayüz prototiplemenin amacı nedir? Yararlı ve aksak yönleriyle belirtiniz.

13. VAD, süreç tanımlama dili ve nesne ilişki şemaları kullanılarak yapılan bir modellemenin, fiziksel tasarıma nasıl yardımcı olacağını açıklayınız.

14. Tasarımla çözümleme arasındaki ilişkiyi belirtiniz.

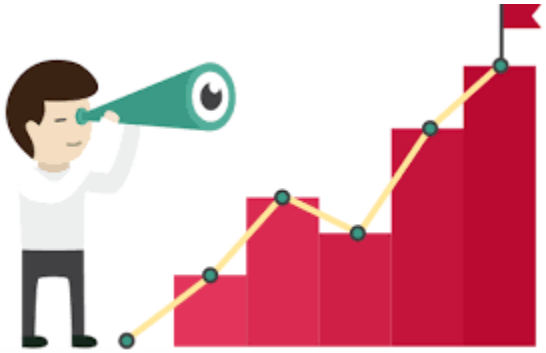
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

6.Hafta

TASARIM

Bölüm Hedefi

- Bu bölümde, yapısal sistem geliştirme metodolojisine göre yapılan bir çözümlemenin tasarıma dönüştürülmesi, tasarım kavramları, veri tasarım ve süreç tasarım yöntemleri açıklanmaktadır. Her tür yazılım için gerekecek ortak alt sistemler tanıtılmaktadır.



Tasarıma Giriş

- Tasarım, Sistem Analizi çalışması sonucunda üretilen Mantıksal Modelin Fiziksel Modele dönüştürülme çalışmasıdır.

- Fiziksel Model geliştirilecek yazılımın;
  - hangi parçalardan oluşacağını,
  - bu parçalar arasındaki ilişkilerin neler olacağını,
  - parçaların iç yapısının ayrıntılarını,
  - gerekecek veri yapısının fiziksel biçimini (dosya, veri tabanı, hash tablosu, vektör, vs)

  tasarımını içerir.

Çözümleme-Tasarım İlişkisi

Kuramsal

Teknik

SRS→SAD

Tasarım Kavramları

- Soyutlama (abstraction): Detayları gizleyerek yukarıdan bakabilme imkanı sağlanır.

- İyileştirme (enhancement): Soyutlama düzeyinde irdeleme bittikten sonra, daha alt seviyelere inilerek tanımlamalarda ayrıntı, bazen de düzeltme yapılarak tasarımın daha kesinlik kazanması sağlanır.

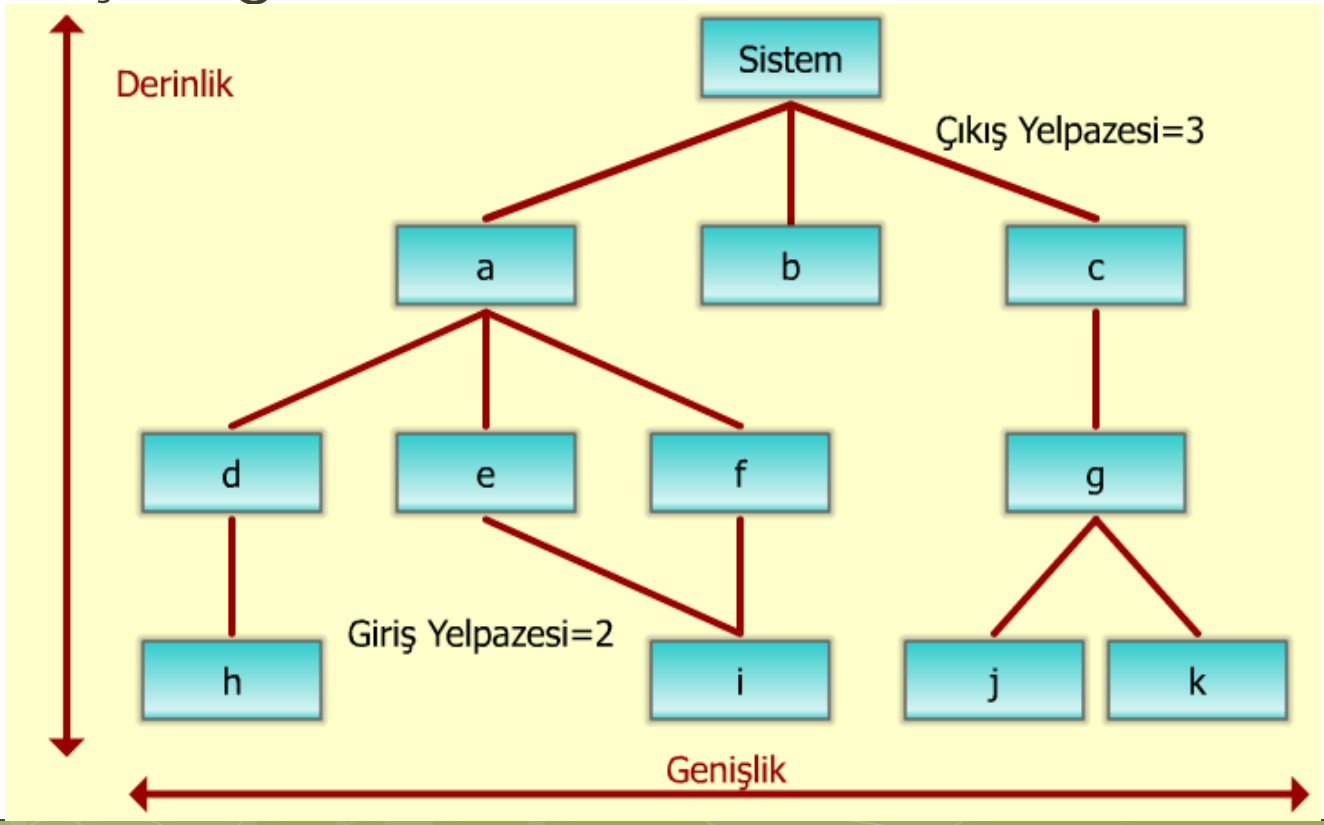
- Modülerlik (modularity): Sistemi istenen kalite faktörleri ışığında parçalara ayrıştırma sonucu elde edilir.

Modülerlik

- Bütün karmaşıklığın tek bir modülde toplanması yerine, anlaşılabilir ve dolayısıyla projenin zihinsel kontrol altında tutulması için sistem bir çok modüle ayrılır.

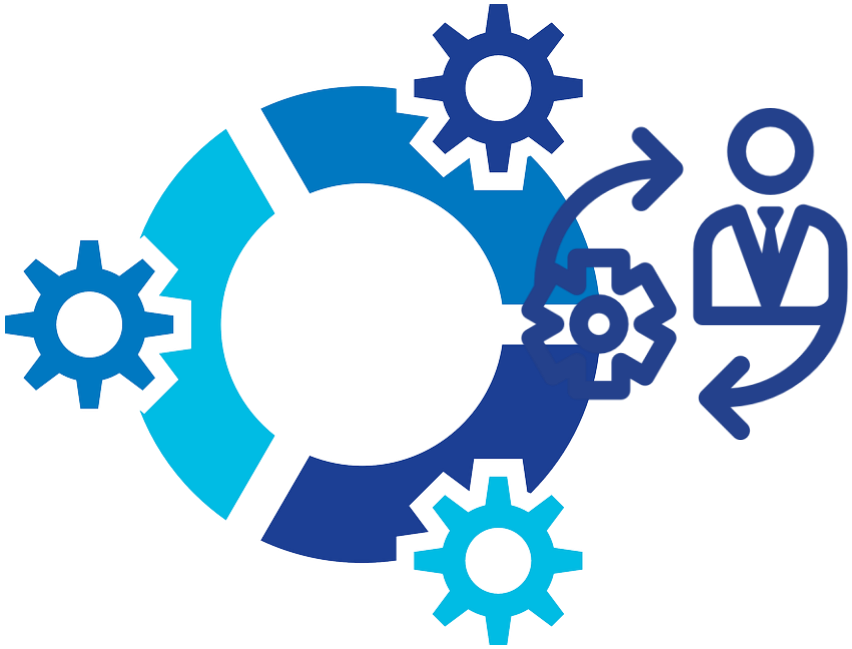
- Modüller, isimleri olan tanımlanmış işlevleri bulunan ve hedef sistemi gerçekleştirmek üzere tümleştirilen birimlerdir.

- Şekil sistemin modüllerden oluşmasını simgelemektedir. Aynı zamanda, sistemin '**kontrol hiyerarşisi**'ni de belirtmektedir. Bir modülün hangi modülleri kullanarak işlevini gerçekleştirdiği bu hiyerarşide görünmektedir.



İşlevsel Bağımsızlık

- Modüllere parametre ile veri gönderilir ve sonuç değer alınır. Bu modülü çağıran program parçası sadece bu sonucu kullanabilir. Çağrılan modülün işlevsel olarak yaptıkları ile ilgili değildir.



Veri Tasarımı

- Yapı Tasarımı, arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.
- Bilgi saklama ve soyutlama bu işlem için önemli kavramlardır.

Veri tasarımı yaparken dikkat edilecek konular

- Değişik veri yapıları değerlendirilmelidir.
- Bütün veri yapıları ve bunlar üzerinde yapılacak işlemler tanımlanmalıdır.
- Alt düzeyde tasarım kararları tasarım süreci içerisinde geciktirilmelidir.
- Bazı çok kullanılan veri yapıları için bir kütüphane oluşturulmalıdır.
- Kullanılacak programlama dili soyut veri tiplerini desteklemelidir.

Yapısal Tasarım

- Yapısal Tasarımın ana hedefi modüler bir yapı geliştirip modüller arasındaki kontrol ilişkilerini temsil etmektir.
- Ayrıca yapısal tasarım bazen de veri akışlarını gösteren biçime dönüştürülebilir.
- Veri Akışları Üç parçada incelenebilir
  - Girdi Akışı
  - Çıktı Akışı
  - İşlem Akışı

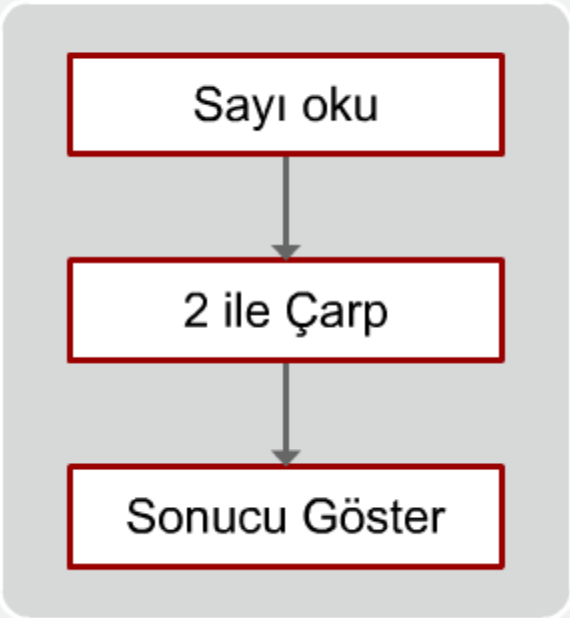
Ayrıntı Tasarım- Süreç Tasarımı

- **Süreç tasarımı;** veri, yapı ve arayüz tasarımından sonra yapılır.
- İdeal şartlarda bütün algoritmik detayın belirtilmesi amaçlanır.
- Ayrıca süreç belirtiminin tek anlamı olması gerekir, değişik şahıslar tarafından farklı yorumlanmamalıdır.
- Doğal diller kullanılabilir (açıklamalarda, çünkü doğal dil tek anlamlı değildir)
- Süreç Tanımlama Dili (PDL)

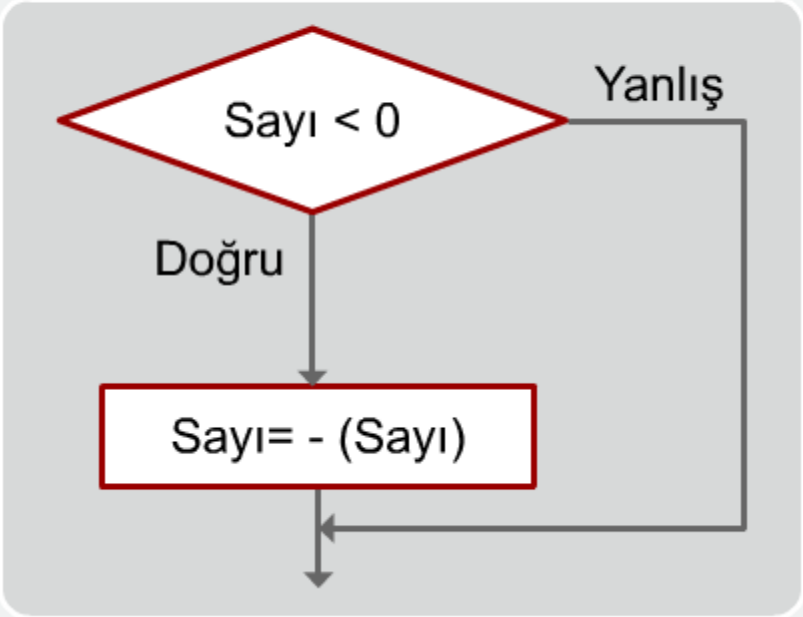
Yapısal Program Yapıları

- Yapısal programlamanın temel amacı;
  - program karmaşıklığını en aza indirmek,
  - program anlaşılabilirliğini artırmaktır.
- Bu amaçla şu yapıları kullanılır;
  - Ardışıl İşlem yapısı
  - Koşullu işlem yapısı
  - Döngü yapısı
- GOTO kullanımı uygun değildir.

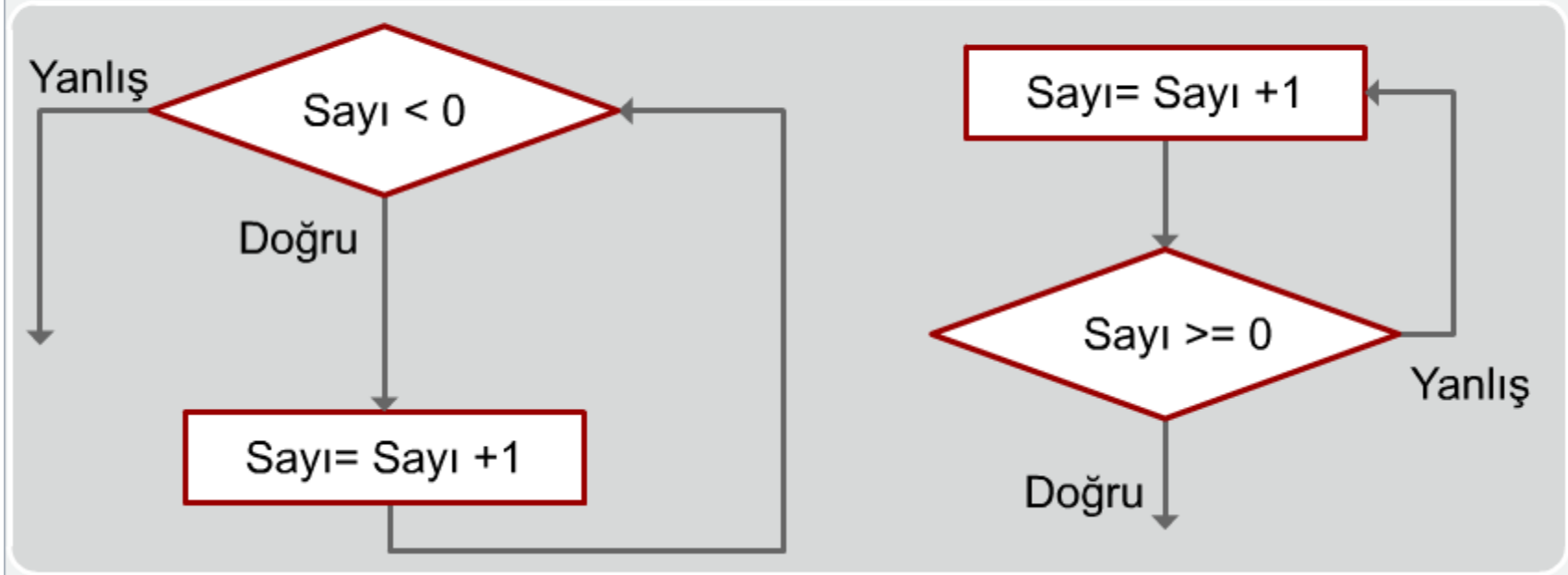
Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Ardışıl İşlem Yapısı | Şartlı İşlem Yapısı | Tekrarlı İşlem Yapıları



Karar Tabloları

- Bazen karmaşık koşul değerlendirmeleri yapmak gerekir. Bunların düzenli bir gösterilimi karar tablolarında yapılabilir.
- Öncelikle, bütün işlemler saptanmalı, sonra ön koşullar belirlenmelidir.
- Belirli işlemler ile belirli koşulları birleştirerek tablo oluşturulur.
- Alt tarafta ise işlemler benzer satırlar olarak gösterilir.

Karar Tablosu Örneği

Kurallar	1	2	3	4
Hesap Geçerli	X	X	X	X
Şifre Geçerli	X		X	X
Yeterli Nakit Var	X			
Yeterli Bakiye Var	X		X	
Bakiye Bildirme İşlemi				X
Para Çekme İşlemi	X			
Para Yatırma İşlemi		X		
Para Gönderme İşlemi			X	

Program Tanımlama Dili

- **Program Tanımlama Dilleri** süreç belirtiminde doğal dillerin programlama dili ile sentezlenmesi şeklinde ortaya çıkmıştır.

- Hangi programlama dilinin kullanılacağından bağımsız özellikler bulunmalıdır.

```
DO
  Hesap Numarasını Oku
  IF (hesap numarası geçerli değil) başlangıca dön
     işlem türünü iste
  IF (para yatırma islemi) { para_yatir(); Başlangıca dön}
  IF (yeterli bakiye yok) başlangıca dön
WHILE
```

Tasarlanması Gereken Ortak Alt Sistemler

- Yetkilendirme altsistemi
- Güvenlik altsistemi
- Yedekleme altsistemi
- Veri transferi altsistemi
- Arşiv altsistemi
- Dönüştürme altsistemi

Yetkilendirme Alt Sistemi

- Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.
  - İşlev bazında yetkilendirme
  - Ekran bazında yetkilendirme
  - Ekran alanları bazında yetkilendirme
- Oracle veri tabanına erişim konusunda yetkilendirme yapmaktadır.

Güvenlik Alt Sistemi

- Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması işlemleri.

- LOG files (Sistem günlüğü)

Yedekleme Alt Sistemi

- Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları veri tabanı (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir.

Veri İletişim Alt Sistemi

- Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleri
- Çevirim içi veri iletimi (real-time)
- Çevirim dışı veri iletimi (disketler, teypler)

Arşiv Alt Sistemi

- Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemlerdir.

- Aktif veri tabanı.

Dönüştürme Alt Sistemi

- Geliştirilen bilgi sisteminin uygulamaya alınmadan önce veri dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması) işlemlerine ihtiyaç vardır.

Kullanıcı Arayüz Tasarımı

- Kullanıcı ile ilişkisi olmayan arayüzler
  - Modüller arası arayüz
  - Sistem ile dış nesneler arası arayüz

- Kullanıcı arayüzleri
  - Kullanım kolaylığı ve öğrenim zamanı esastır
  - Program=arayüz yaklaşımı vardır

Genel Prensipler

- Veri giriş formlarının tutarlı olması
- Önemli silmelerde teyit alınmalı
- Yapılan çoğu işlem geri alınabilmeli
- Hataların affedilmesi, yanlış girişte kırılmama
- Komut isimlerinin kısa ve basit olması
- Menülerin ve diğer etkileşimli araçların standart yapıda kullanımı

Bilgi Gösterimi

- Yalnızca içinde bulunulan konu çerçevesi ile ilgili bilgi gösterilmeli
- Veri çokluğu ile kullanıcı bunaltılmamalı, grafik ve resimler kullanılmalı
- Tutarlı başlık, renkleme ve kısaltma kullanılmalı
- Hata mesajları açıklayıcı ve anlaşılır olmalı
- Değişik tür bilgiler kendi içinde sınıflandırılmalı
- Rakamsal ifadelerde analog görüntü verilmeli (%89 değil)

Veri Girişi

- Kullanıcı hareketleri en aza indirilmeli
- Gösterim ve girdi sahaları birbirinden ayrılmalı (renk)
- Kullanıcı uyarlamasına izin verilmeli, kullanıcı bazı özellikleri tanımlayabilmeli
- Kullanılan konu ile ilgili gereksiz komutlar deaktifleştirilmeli
- Bütün girdiler için yardım kolaylıkları olmalı

Kullanıcı Arayüz Prototipi

- Tasarım çalışması sonucunda, daha önceden gereksinim çalışması sırasında hazırlanmış olan kullanıcı arayüz prototipi, ekran ve rapor tasarımları biçimine dönüşür. Ekranlar son halini alır, raporlar kesinleşir. Kullanıcıya gösterilerek onay alınır.

- Tüm programın tek elden çıktığının ifade edilebilmesi açısından tüm ekranların aynı şablon üzerine oturtulması önerilmektedir.
  - **Menü Çubuğu**
  - **Araç Çubuğu**
  - **Gövde (Değişebilir)**
  - **Durum Çubuğu**

Başlangıç Tasarım Gözden Geçirme

- Yapılan tasarım çalışmasının bir önceki geliştirme aşaması olan analiz aşamasında belirlenen gereksinimleri karşılayıp karşılamadığının belirlenmesidir.
  - Sistem gereksinimlerine yardımcı olan kullanıcılar
  - Sistem analizini yapan çözümleyiciler
  - Sistemin kullanıcıları
  - Tasarımcılar
  - Yönlendirici
  - Sekreter
  - Sistemi geliştirecek programcılar

  dan oluşan bir grup tarafından yapılır.

Ayrıntılı Tasarım Gözden Geçirme

- Başlangıç tasarımı gözden geçirme çalışmasının başarılı bir biçimde tamamlanmasından sonra, tasarımın teknik uygunluğunu belirlemek için Ayrıntılı Tasarım Gözden Geçirme çalışması yapılır. Bu çalışmada;
  - Çözümleyiciler
  - Sistem Tasarımcıları
  - Sistem Geliştiriciler
  - Sekreter

  den oluşan bir ekip kullanılır.

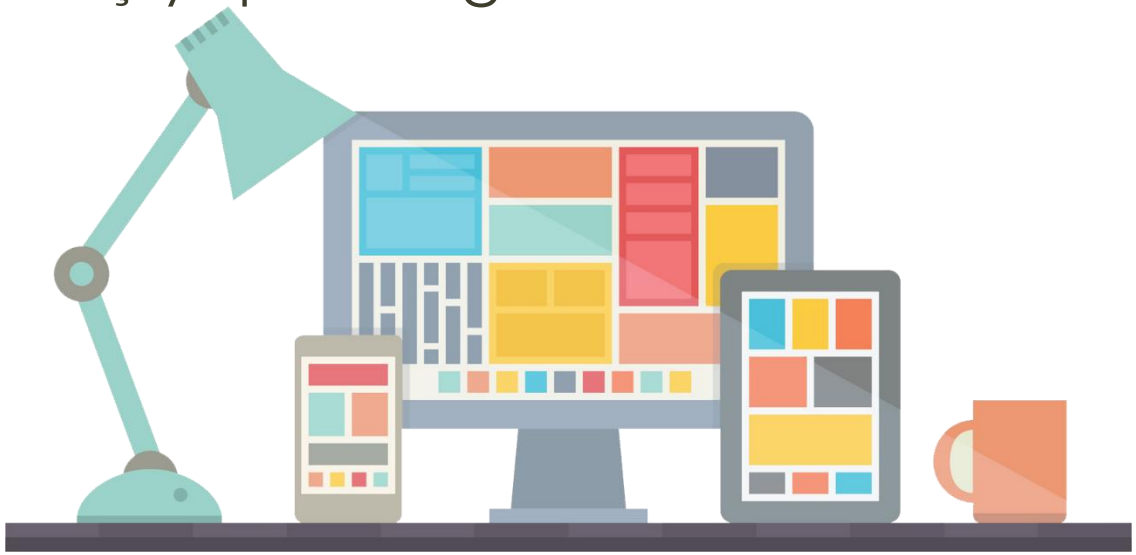
Tasarım Kalite Ölçütleri

- Bağlaşım (Coupling)

  Tasarımı oluşturan modüller arası ilişki ile ilgilidir.

- Yapışıklık (Cohesion)

  Modüllerin iç yapısı ile ilgilidir.



Bağlaşım

- Modüller arası bağlılığın ölçülmesi için kullanılan bir ölçüttür.
- Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır.
- Bağlaşımın düşük olması
  - Hatanın dalgasal yayılma özelliğinin azaltılması
  - Modüllerin bakım kolaylığı
  - Modüller arası ilişkilerde karmaşıklığın azaltılması

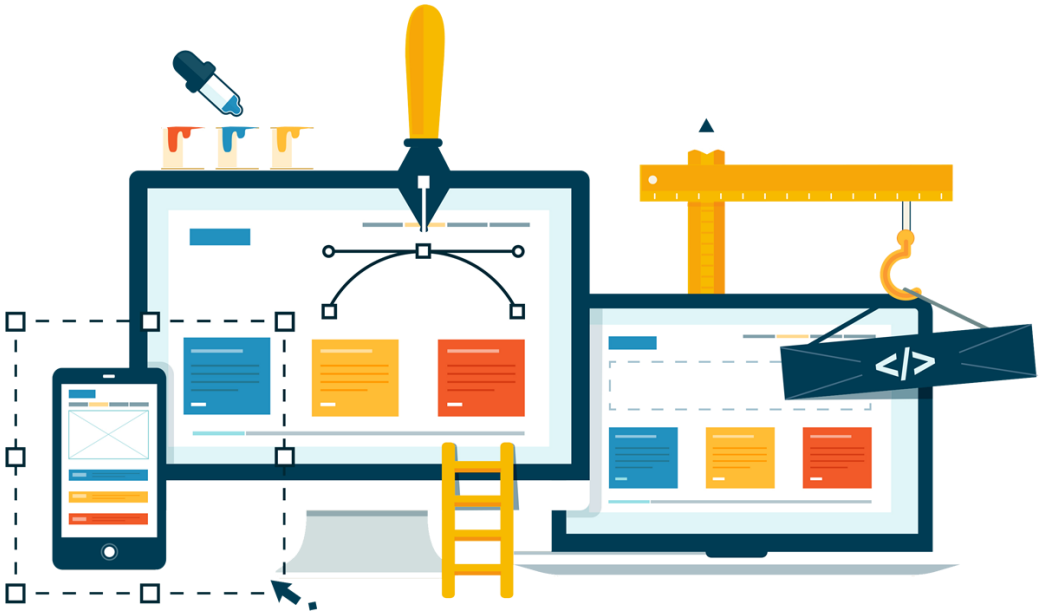
  nedenleri ile istenmektedir

Yalın Veri Bağlaşımı

- Herhangi iki modül arası iletişim yalın veriler (tamsayı, karakter, boolean, vs) aracılığı ile gerçekleştiriliyorsa bu iki modül yalın veri bağlaşımlıdır şeklinde tanımlanır.

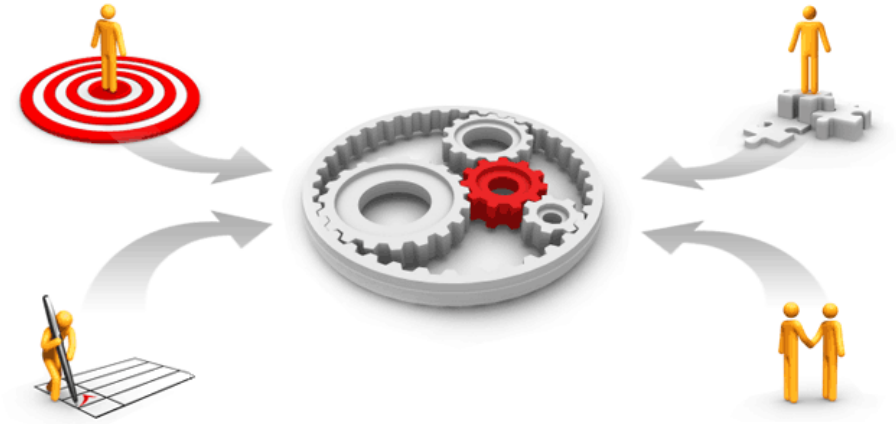
Karmaşık Veri Bağlaşımı

- Herhangi iki modül arasındaki iletişimde kullanılan parametrelerin karmaşık veri yapısı (kayıt, dizi, nesne, vs) olması durumunda modüller karmaşık veri paylaşımlı olarak tanımlanır.



Denetim Bağlaşımı

- İki Modül arasında iletişim parametresi olarak denetim verisi kullanılıyorsa bu iki modül denetim bağlaşımlı olarak tanımlanır.



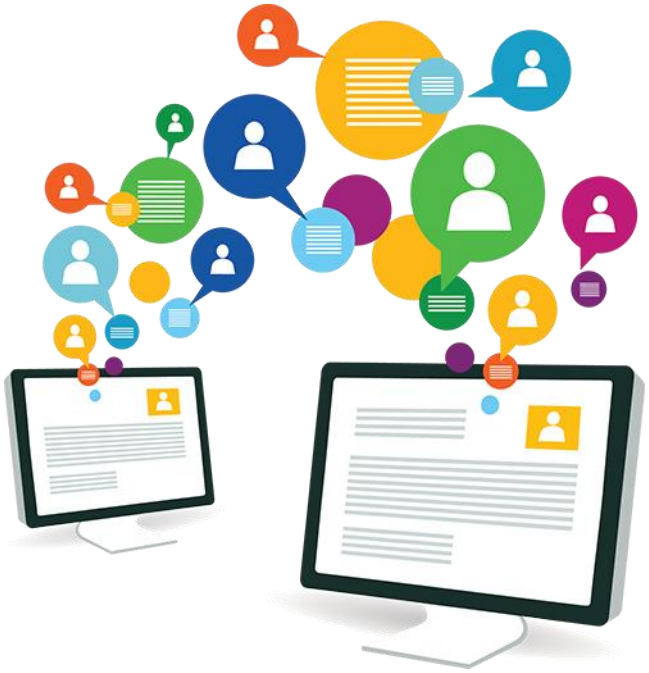
Ortak Veri Bağlaşımı

- Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül ortak veri bağlaşımlı olarak tanımlanır.

- Verilerin ortak veri bağlaşımlı olmaları şu nedenlerden dolayı fazla istenmez;
  - Ortak veri alanını izlemek zordur.
  - Ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler.
  - Ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanacak bütün modüller göz önüne alınmalıdır.

İçerik Bağlaşımı

- Modüllerin iç içe tasarlanması sonucu, bir modülün başka bir modül içerisinde tanımlanmış veri alanına erişebilmesi olanaklaşır ve bu durum içerik bağlaşımına yol açar.



Yapışıklık

- Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir ölçüttür. Modül gücü olarak ta tanımlanır.
- Tasarımda yapışıklık özelliğinin yüksek olması tercih edilir.
- Yapışıklık ile Bağlaşım ters orantılıdır.

İşlevsel Yapışıklık

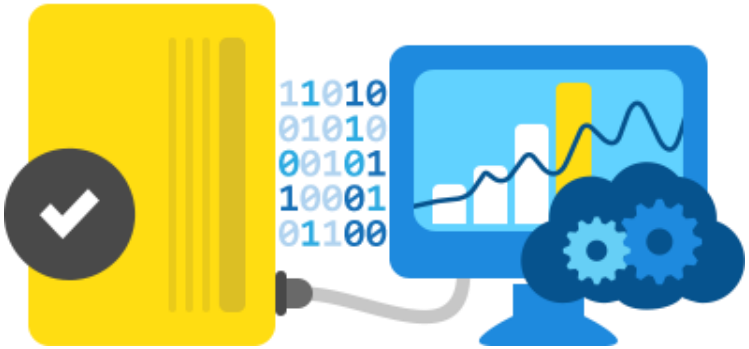
- İşlevsel Yapışık bir modül, tek bir iş problemine ilişkin sorunu çözen modül olarak tanımlanır.

- Maas_Hesapla, Alan_Hesapla gibi

Sırasal Yapışıklık

- Bir modülün içindeki işlemler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanılıyorsa bu modül sırasal yapışık bir modül olarak adlandırılır.

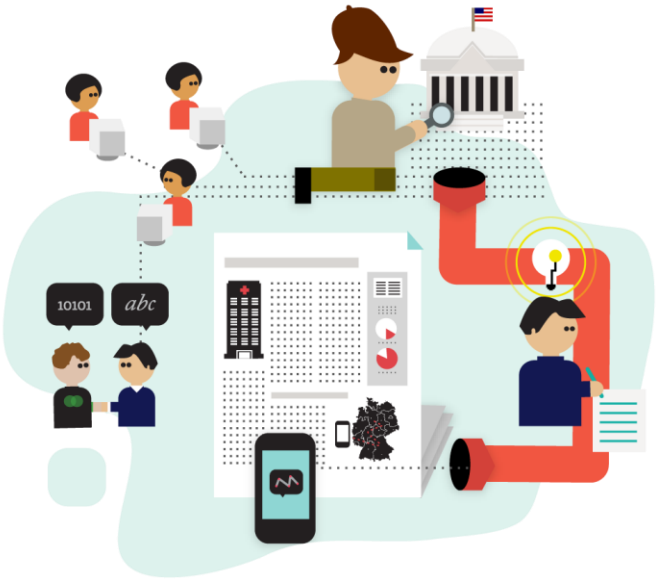
Ham_Veri_Kaydını_Düzelt
Duzeltilmis_Ham_Veri_Kaydini_Dogrula
Dogrulanmis_Kaydi_Gonder



İletişimsel Yapışıklık

- Bir modülün içindeki farklı işlemler aynı girdi ya da çıktıyı kullanıyorlarsa bu modül iletişimsel yapışık bir modül olarak adlandırılır.

Sicil_No_yu_Al
Adres_Bilgisini_Bul
Telefon_Bilgisini_Bul
Maas_Bilgisini_Bul



Yordamsal Yapışıklık

- Yordamsal Yapışık modüldeki işlemler arasında denetim ilişkisi bulunmaktadır.
- İşlemlerin birbirleri ile veri ilişkisi yoktur, ancak işlem sırası önemlidir.

Ekran_Goruntusunu_Yaz

Giris_Kaydini_Oku

Zamansal Yapışıklık

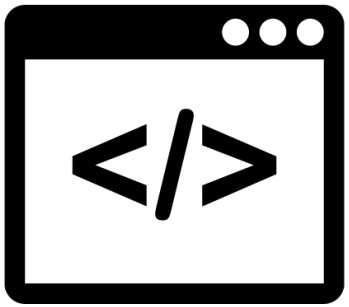
- Bir modül içindeki işlemlerin belirli bir zamanda uygulanması gerekiyor ve bu işlemlerin kendi aralarında herhangi bir ilişkisi yok, yani işlemlerin sırası önemli değil ise, zamansal yapışıklık vardır.

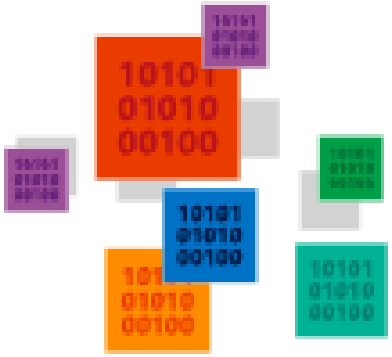
Alarm_Zilini_Ac
Kapiyi_Ac
Kamerayi_Calistir

Mantıksal Yapışıklık

Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller mantıksal yapışık olarak adlandırılır.

Dizilere değer atama işlemleri

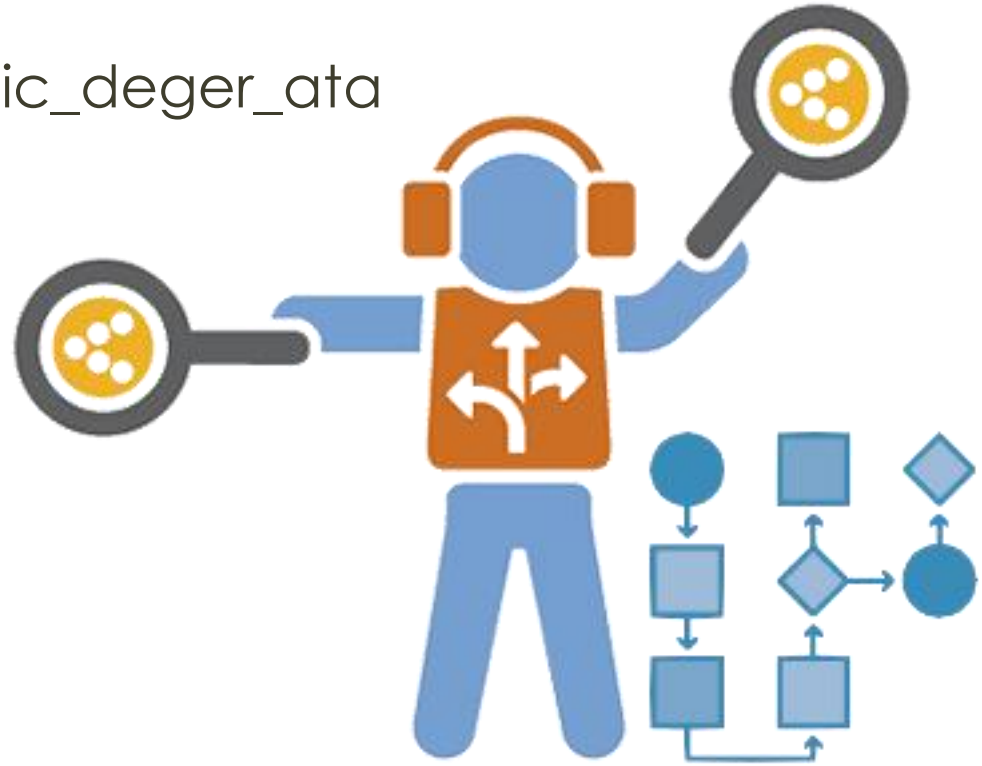




Gelişigüzel Yapışıklık

- İşlemler arasında herhangi bir ilişki bulunmaz.

Ara_Kayit_Oku
B_dizisine_baslangic_deger_ata
Stok_kutugu_oku
Hata_iletisi_yaz

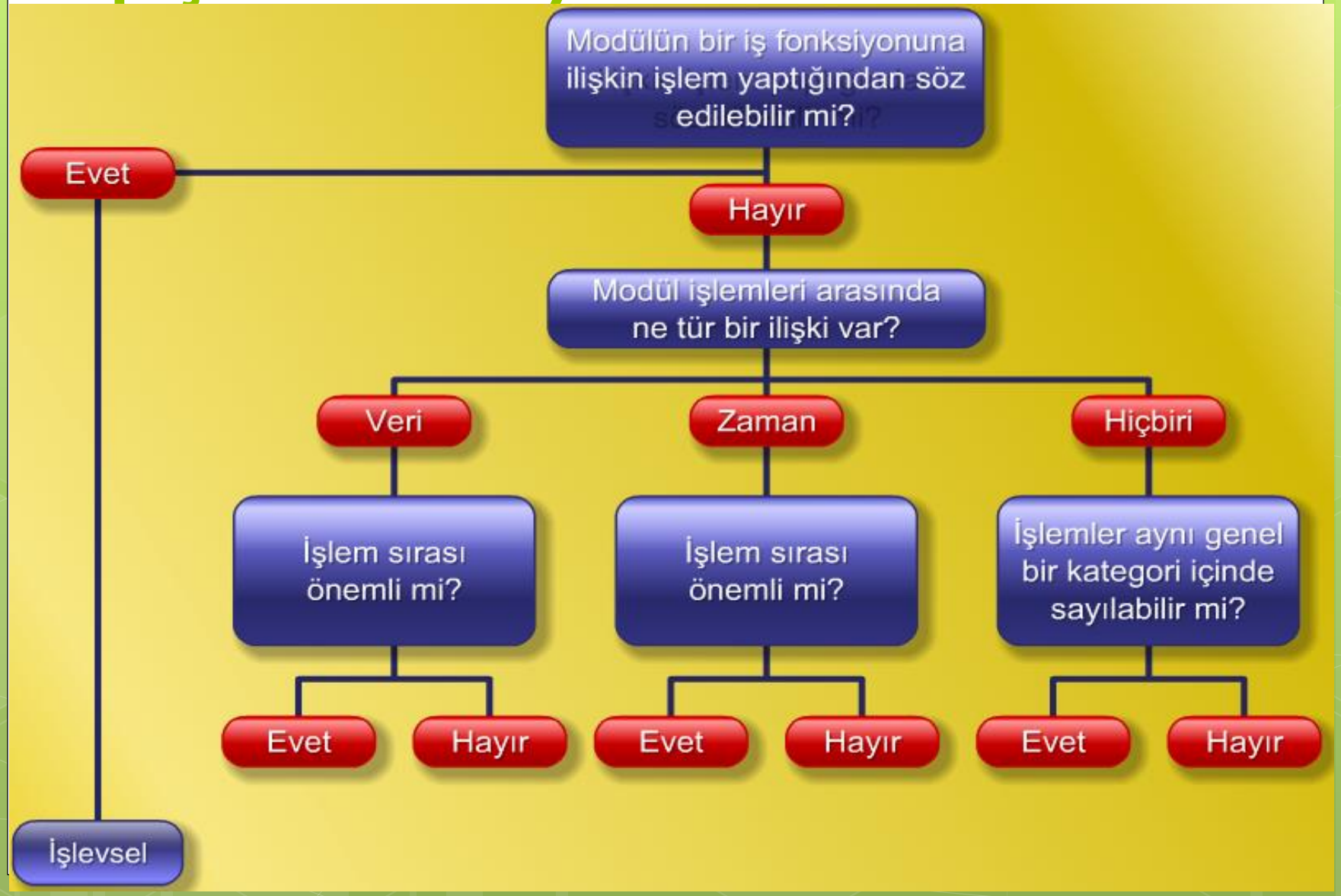


Yapışıklık Hesaplama

- Genel olarak bir modülün yapışıklık düzeyini hesaplamada bazı güçlüklerle karşılaşılabilir. Çünkü, bir modüldeki işlemler hep aynı tür özellik göstermeyebilir.

Yapışıklık Türü	Ölçek	Kalite
İşlevsel	1	En iyi
Sırasal	2	
İletişimsel	3	
Yordamsal	4	
Zamansal	5	
Mantıksal	6	
Gelişigüzel	7	En kötü

Yapışıklık Düzeyine Karar Verilmesi

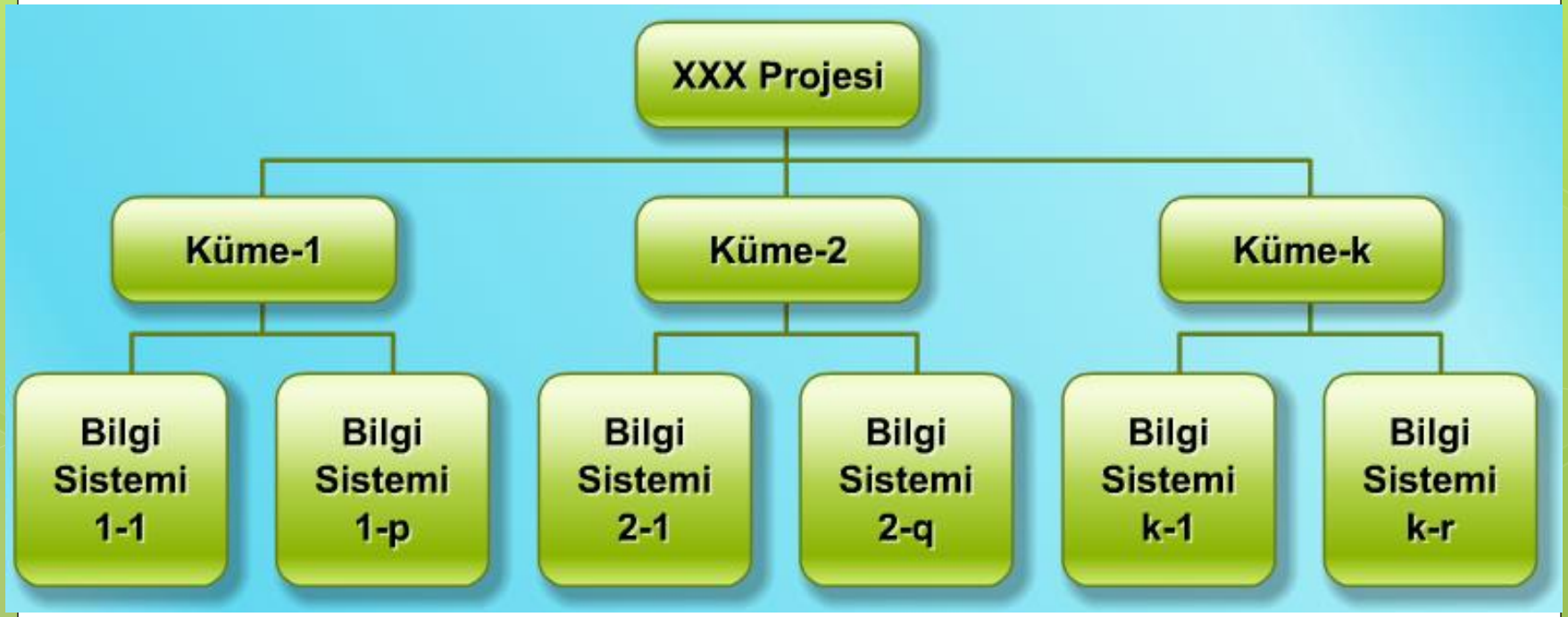


TASARIM RAPORU İÇERİĞİ

- İçeriğin hazırlanmasında IEEE/ANSI standartları kullanılmıştır. Büyük boyutlu projeler için burada verilen içerik, genel mantık aynı kalmak koşulu ile detaylandırılabilir. Örneğin 200 kişi-yıllık işgücü gerektiren bir BT projesi, Şekil'deki gibi bölümlenebilir.

- Ödev IEEE 1016-1998

Tasarım Raporu İçeriği



Örnek Tasarım Raporu İçeriği

1. Giriş

1.1. Kapsam
1.2. Tanımlar, Kısa Adlar ve Kısaltmalar
1.3. Referanslar
1.4. Belgenin Yapısı

2. Genel Tasarım Bilgileri

2.1. Genel Sistem Tanımı
2.2. Varsayımlar ve Kısıtlamalar
2.3. Sistem Mimarisi
2.4. Dış Arabirimler

2.4.1. Kullanıcı Arabirimleri
2.4.2. Veri Arabirimleri
2.4.3. Diğer Sistemlerle Arabirimler

2.5. Veri Modeli
2.6. Testler
2.7. Performans

Örnek Tasarım Raporu İçeriği

3. Veri Tasarımı

3.1. Tablo tanımları
3.2. Tablo- İlişki Şemaları
3.2. Veri Tanımları
3.3. Değer Kümesi Tanımları

4. Süreç Tasarımı

4.1. Genel Tasarım
4.2. Modüller

4.2.1. XXX Modülü

4.2.1.1. İşlev
4.2.1.2. Kullanıcı Arabirimi
4.2.1.3. Modül Tanımı
4.2.1.4. Modül iç Tasarımı

4.2.2. YYY Modülü

4.3. Kullanıcı Profilleri
4.4. Entegrasyon ve Test Gereksinimleri

Örnek Tasarım Raporu İçeriği

5. Ortak Alt Sistemlerin Tasarımı

5.1. Ortak Alt Sistemler
5.2. Modüller arası Ortak Veriler
5.3. Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri
5.4. Güvenlik Altsistemi
5.5. Veri Dağıtım Altsistemi
5.6. Yedekleme ve Arşivleme İşlemleri

EKLER

EK-A Rapor Görünümleri
EK-B Test Belirtimleri
EK-C Kullanıcı Kılavuzu Örneği

Alınan Dersler 1

- Ortak alt sistemler oldukça önemlidir. Tasarlanması ve geliştirilmesi için ayrı bir yazılım ekibi kullanılması gerekir. Aksi durumda, tutarsızlık ve işlem tekrarı gündeme gelir, işgücü kaybına neden olunur.

Alınan Dersler 2

- Özellikle, coğrafik olarak birden çok alana yayılmış kuruluşlar için geliştirilecek bilgi sistemlerinde gerek merkez gerekse bağlı birimlerde çalışacak yazılımın aynı yazılım olarak tasarlanıp geliştirilmesi uygulama ve bakım kolaylığı açısından önerilmektedir.

Alınan Dersler

- Genellikle bu tür uygulamalarda, merkezdeki yazılım, birimlerdeki yazılımlara oranla daha farklı unsurlar içermekte, ancak birimlerdeki uygulamaları da kapsamaktadır.
- Bu durumda, neden merkezdeki yazılımın işlevlerini, kullanılmayacaklarını bile bile birimlere de taşıyalım sorusu akla gelebilir.

Alınan Dersler

- Zira bu durum kaynak kullanımı açısından birimlere ek yük getirmektedir. Bu durumda bakım ve kurulum kolaylığı ve fazla kaynak kullanımı arasında bir tercih yapmak gerekmektedir.
- Günümüzde, bilgisayar kaynaklarının giderek ucuzladığı düşünüldüğünde bu tercihin, bakım ve kurulum kolaylığı yönünde yapılması önerilmektedir.

Alınan Dersler 3

- Genellikle kurumsal BT projelerinde veri iletişiminin tüm türlerine (Çevrim-içi anında, çevrim-içi kargo, çevrim dışı) rastlanmaktadır. Bu durumda, Veri İletişim alt sistemlerinin bu özellikleri içerecek biçimde tasarlanması gerekmektedir.
- Özellikle çevrim içi ve anında iletişim gerektiren durumlarda, işlemlerin izlenmesi ve herhangi bir aksaklık durumunda geri alınması gibi işlevlerin tasarımda içerilmesi gerekmektedir.

Alınan Dersler 4

- Dönüştürme işlemleri, yazılım gereksinim tanımlarında oldukça az ve yetersiz biçimde tanımlanmakta ve bu durum yüklenici ile iş sahibi arasında sorunlara yol açmaktadır.
- Geçmiş uygulamalarda, bilgisayar ortamında saklanan bilgilerin yeni uygulama ortamında oluşturulacak veri tabanına aktarılmasında ortaya çıkan temel sorun, geçmiş verilerdeki tutarsızlıklardır.

Alınan Dersler

- Bir taraftan bu tür tutarsızlıkların yeni sistemde bulunmaması öngörülürken, öte yandan tutarsızlık içeren geçmiş verilerin aktarılması istenir. Bu durum tasarımda içerilmemişse büyük zorluklarla karşılaşılır.
- Örneğin, nüfus uygulamasındaki kişilerin evlilikleri ile ilgili bağ alanlarının dolu olması bir zorunluluk olarak öngörüldüğünü varsayalım.

Alınan Dersler

- Öte yandan, bilgisayar ortamındaki mevcut verilere bakıldığında bir çok bağ alanının boş olduğu görüldüğünde mevcut verilerin aktarımında zorluklar çıkar.
- Veri tabanındaki kısıtlar kaldırılıp veriler olduğu gibi aktarılsa bile, bu verilere ilişkin ileride ortaya çıkabilecek işlemlerde yine sorunlar oluşabilir.

Alınan Dersler

- Bu tür durumların, yazılım gereksinim tanımlarında ayrıntılı olarak tanımlanması, tasarım sırasında dikkatlice ele alınmalarını sağlar.

Sorular

1. Yazılım tasarım sürecinin temel işlemlerini sayınız. E-R diyagramı çizerek ilişkilerini gösteriniz.

2. Geliştireceğiniz bir uygulama için ekran şablonunuzu belirleyiniz.

3. Bağlaşım ve yapışıklık kavramlarını açıklayınız. Program bakımı ile ilişkilerini belirtiniz.

4. Bir sistem tümüyle bağlaşımsız biçimde tasarlanabilir mi? Yani sistemin tüm modülleri arasında hiç bağlaşım olmadan tasarım yapılabilir mi?

5. Tümüyle işlevsel yapışık modüllerden oluşan bir sistem tasarlanabilir mi? Neden yapılabilir? Neden yapılamaz?

6. Bağlaşım ile Yazılım Taşınabilriliği arasındaki ilişkiyi belirtiniz.

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

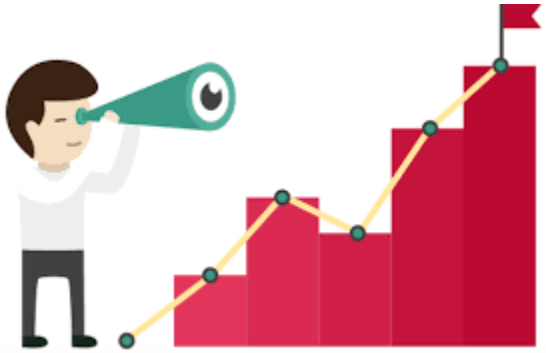
YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ

7.Hafta

GERÇEKLEŞTİRME

Bölüm Hedefi

- Bu bölümde, yazılım geliştirme ortamları tanıtılmakta ve kodlama yöntemleri açıklanmaktadır. Program karmaşıklığının ölçümü üzerine yaygın olarak kullanılan yöntemlerden biri olan Mc Cabe karmaşıklık ölçütü ve kod gözden geçirme teknikleri bu bölümde tanıtılmaktadır.



Gerçekleştirme Fazı

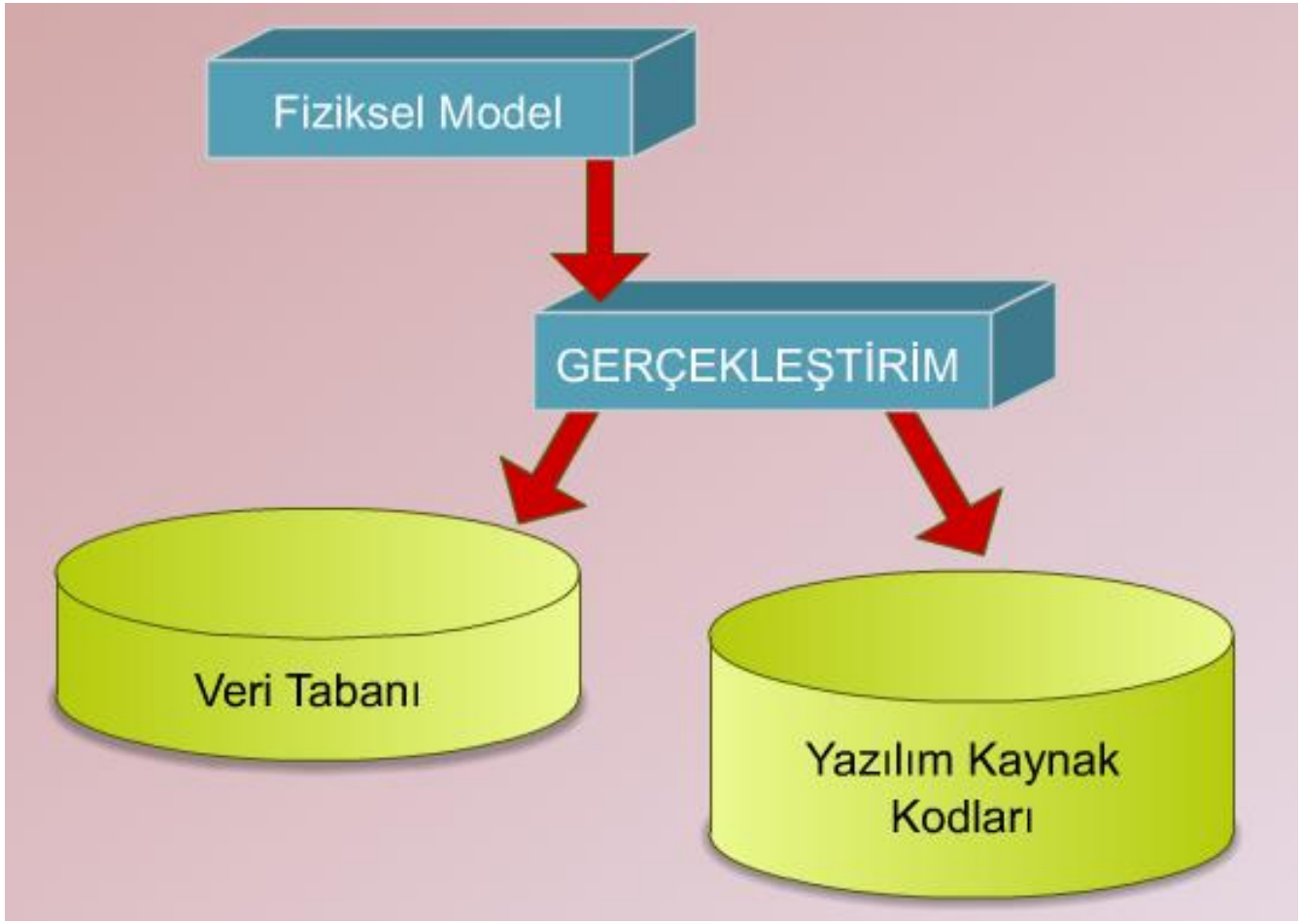
- Programlama Dilleri
- Yazılım Gerçekleştirme
- Derleyiciler
- Kodlama Etkinliği ve Kuralları
- Modüler Gerçekleştirme
- Gerçekleştirmenin Belgelendirilmesi

Giriş

- Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir.

- Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

Gerçekleştirim Çalışması



Yazılım Geliştirme Ortamları

- Programlama Dili
- Veri Tabanı Yönetim Sistemi
- Hazır Program Kitapçıkları
- CASE Araçları

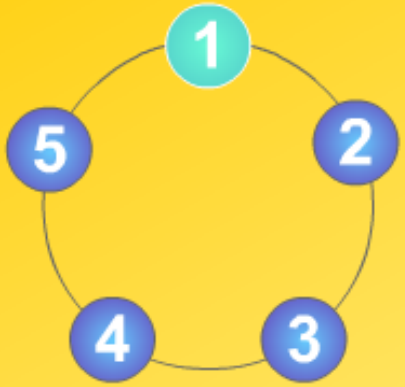
Programlama Dilleri

Geliştirilecek uygulamaya göre kullanılabilecek programlama dili de değişmektedir. Uygulama alanlarının bir sınıflaması:

- Veri işleme yoğunluklu uygulamalar
- Hesaplama yoğunluklu uygulamalar
- Süreç ağırlıklı uygulamalar
- Sistem programlamaya yönelik uygulamalar
- Yapay us (Y. zekâ- Y.Akıl) uygulamaları

biçiminde verilmektedir.

Uygulama Çeşitleri

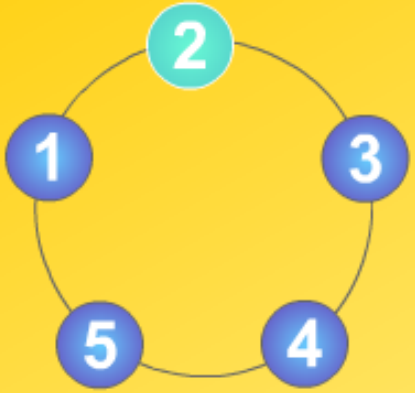


1

Veri İşleme Yoğunluklu Uygulamalar

Günümüzde, bilgi sistemi uygulamaları olarak bilinen ve yoğun veri işlemeyi gerektiren uygulamalar (Nüfus, Seçmen Kütükleri, Adli Sicil vb) için 90'lı yıllara kadar COBOL en yaygın olarak kullanılan programlama dili idi. 90'lı yılların başlarında gelişen ve veri tabanı bağlantılı çalışan Görsel Programlama Dilleri (Visual Basic, Delphi, Visual Age, Developer vb) bugünün uygulamalarında yaygın olarak kullanılmakta ve COBOL ve geçmişteki uygulamalar da bu dillere dönüştürülmektedir. Öte yandan Nesne-Kökenli Programlama dillerinin (C++, Java, Smalltalk vb) kullanımı giderek yaygınlaşmaktadır.

Uygulama Çeşitleri



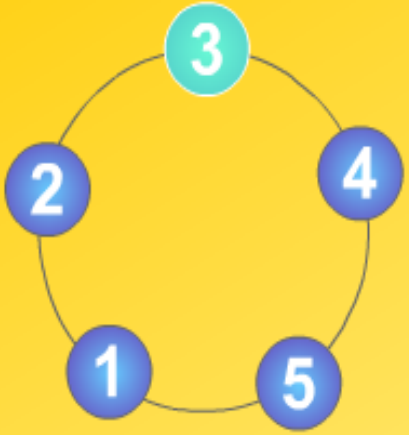
2

Yoğun Matematiksel ve Mantık İşlemi Gerektiren Uygulamalar

Bu uygulamalarda (NASA, uçak mühendisliği vb) program kütüphanelerinin zenginliği nedeniyle halen en yaygın olarak kullanılan programlama dilleri arasında, FORTRAN yer almaktadır. Yeni geliştirilen, bu tür uygulamalarda ağırlıklı olarak C programlama dili kullanılmaktadır.

Teknolojik gelişim, bu tür uygulamaları hızlandırmak amacıyla paralel işlem yapmayı olanaklı kılmış, bu bağlamda Paralel Programlama dilleri gelişmiştir. Ayrıca mevcut programlama dillerinin paralel sürümleri geliştirilmiştir (Paralel FORTRAN, Paralel C vb). Bu alanda kullanılan programlama dillerinin bir başka özelliği ise, yüksek duyarlıkta işlem yapmayı olanaklı kılmalarıdır.

Uygulama Çeşitleri

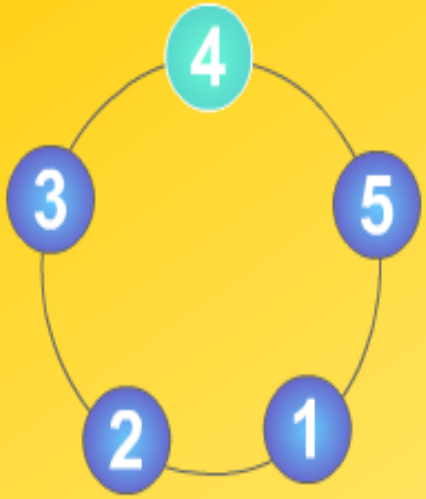


3

Süreç Ağırlıklı Uygulamalar

Bir başka deyişle "gerçek zamanlı" uygulamalarda, veri işleme ve matematiksel işlem yapma boyutlarının yanı sıra süreçler arası eşgüdüm, donanımın yazılım aracıyla sürülmesi boyutları önem kazanmaktadır. Bazı işletim sistemi yazılımları, uçak kontrol sistemleri, bankacılık sistemleri bu tür yazılımlara örnek olarak verilebilir. Bu tür uygulamalarda genellikle düşük düzey programlama dilleri (assembly dili) ve özel programlama dilleri ve henüz yaygınlaşmamış olmakla birlikte C programlama dili kullanılmaktadır.

Uygulama Çeşitleri

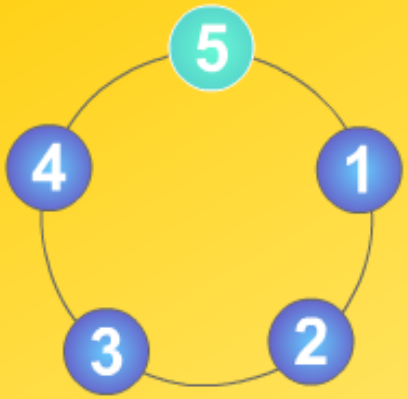


4

Sistem Programlamaya Yönelik Uygulamalar

Bilgisayarın yönetimine ilişkin uygulamalardır (İşletim sistemi vb). Bu tür yazılımlarda kullanılan programlama dili kapalı sistemler için assembly programlama dili, açık sistemler için ise, assembly dili ile birlikte C programlama dilidir.

Uygulama Çeşitleri



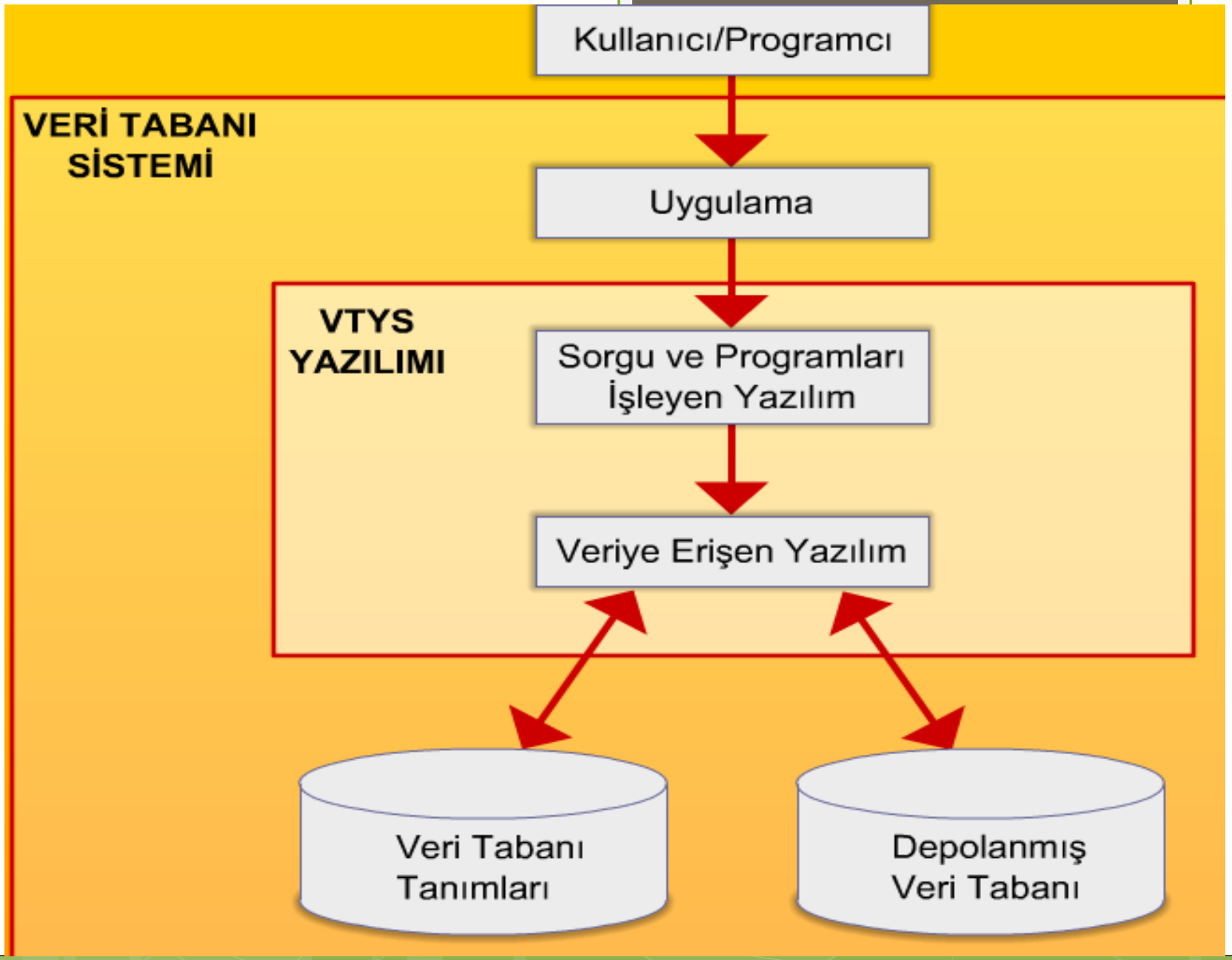
5

Yapay Us Uygulamaları

Kural-çıkarım ilkelerine dayalı uygulamalardır. Kuralların tanımlanması, bu kurallardan çıkarımlar yapılması ve çıkarımlardan yeni kurallar tanımlanması biçiminde etkileşimli olarak çalışmayı gerektiren uygulamalardır. Bu tür uygulamalarda en yaygın olarak kullanılan programlama dilleri ise Lisp ve Prolog'dur.

Veri Tabanı Yönetim Sistemleri

- Birbiri ile ilişkili veriler topluluğu veri tabanı olarak tanımlanmaktadır. Veri tabanı herhangi bir boyutta ya da karmaşıklıkta olabilir.
- Kişisel telefon rehberinizdeki adres bilgileri bir veri tabanı örneği oluşturduğu gibi, maliye bakanlığı bünyesinde saklanan ve vergi ödemesi gereken tüm kişilerin bilgilerinin saklandığı uygulama da bir başka veri tabanı örneğidir.
- Veri tabanını oluşturan veriler birbiriyle ilişkili verilerdir. Bir veritabanında veriler arası ilişkiler ile veri değerleri bulunur.


Kullanıcı/Programcı
VERİ TABANI SİSTEMİ
Uygulama
VTYS YAZILIMI
Sorgu ve Programları İşleyen Yazılım
Veriye Erişen Yazılım
Veri Tabanı Tanımları
Depolanmış Veri Tabanı

Veritabanı Yaklaşımının Geleneksel Kütük Kavramından Ayıran Özellikler

- 1. Veri Sözlüğü
- 2. Veri Soyutlama
- 3. Program-Veri ve Program-İşlem Bağımsızlığı
- 4. Birden Çok Kullanıcı Desteği
- 5. Verinin Birden Fazla İşlem Arasında Paylaşımı

Veri Sözlüğü

- Uygulama yazılımında kullanılan tüm veri tanımlarının yapısal ve ayrık bir biçimde saklanmasını sağlayan bir katalog bilgisi veya **veri sözlüğü**nün varlığı.

Dosya Adı	Değişken Adı	Veri Tipi	Uzunluk	Açıklama
Users	user_id	int	4	Kullanıcı Numarası (ID)
	nick	varchar	10	Kullanıcı Adı
	fullname	varchar	20	Kullanıcı Adı Soyadı
	email	varchar	20	Kullanıcı E-Mail Adresi
	password	varchar	8	Kullanıcı Şifresi
	accesslevel	int	1	Kullanıcı Yetkilendirme Seviyesi (0..9)
Codes	code_id	int	6	Kod Numarası (ID)
	language	varchar	15	Programlama Dil İsmi
	code_category	varchar	15	Kod Kategorisi
	add_date	datetime		Kod Ekleme Tarihi
	score	int	2	Kod Puanı (0-10)
	search_word	varchar	30	Kod Arama Kelimeleri
Forum	comments	longtext		Forum Yorumları
	forum_category	varchar	15	Forum içerisindeki dil kategorileri
	forum_date	datetime		Foruma bilgi giriş tarihi
Code_DB	VT_id	int	2	Veri Tabanı Numarası (ID)
	VT_code_lang	varchar	15	VT'deki Programlama Dilleri
User_DB	user_number	int	4	(Derived) Kullanıcı ID

Veri Soyutlama

- Veri tabanı yaklaşımının bir temel karakteristiği kullanıcıdan verinin saklanması konusundaki detayları gizleyerek veri soyutlamasını sağlamasıdır. Bu soyutlamayı sağlayan ana araç veri modelidir. Veri modeli şu şekilde tanımlanabilir.

Bir veri modeli bir veritabanının yapısını açıklamakta kullanılan kavramlar setidir.

Veri Soyutlama

- Bir veri tabanının yapısı ile veri tipleri, ilişkiler ve sınırlamalar kastedilmektedir. Pek çok veri modeli ayrıca veri tabanı üzerinde belirtilen getiri (çağrı) ve güncellemeler için temel işlemler setini içerir.
- Davranışı belirlemek üzere veri modelinin içine kavramları da eklemek mümkündür. Bu da, temel işlemlere ek olarak kullanıcı tarafından tanımlanmış işlemlerin de veri modeline eklenmesiyle mümkündür.
- Bir veri modelindeki genel işlemlere örnek olarak Ekle, Sil, Değiştir, Eriş verilebilir.

Program-Veri ve Program-İşlem Bağımsızlığı

- Geleneksel dosya işleme yönteminde, veri dosyalarının yapısı bu dosyalara erişim programları içine gömülmüştür. Bundan dolayı da dosyanın yapısındaki herhangi bir değişiklik bu dosyaya erişen tüm programlarda değişiklik yapılmasını gerektirir.
- Tersine VTYS erişim programları veri dosyalarından bağımsız olarak tasarlanmışlardır. VTYS de saklanan Veri dosyalarının yapısı erişim programlarından ayrı olarak katalogda tutulduğundan program ve veri bağımsızlığı sağlanır.

Program-Veri ve Program-İşlem Bağımsızlığı

ÖRNEK

Örneğin bir veri dosyasına yeni bir alan eklenmek istendiğinde, bu veri dosyasını kullanan tüm programlara bu alanın eklenmesi gerekir. VTYS yaklaşımında ise sadece katalogdaki veritabanı tanımlarına bu alan eklenerek bu sorun çözülebilir.

Birden Çok Kullanıcı Desteği

- Bir veri tabanının birçok kullanıcısı vardır ve bunların her biri veri tabanının farklı bir görüntüsüne gereksinim duyabilir. Bir görüntü veri tabanının alt kümesi olabilir veya veri tabanından elde edilmiş fakat kesin olarak saklanmamış sanal veri içerebilir.
- Bazı kullanıcılar, kullandıkları verilerinin veri tabanından türetilmiş veriler mi? Yoksa veri tabanında gerçekte saklanan veriler mi olup olmadığı ile ilgilenmezler. Kullanıcıları farklı uygulamalara sahip çok kullanıcılı bir VTYS çoklu görüntü tanımlama olanaklarını sağlamalıdır.

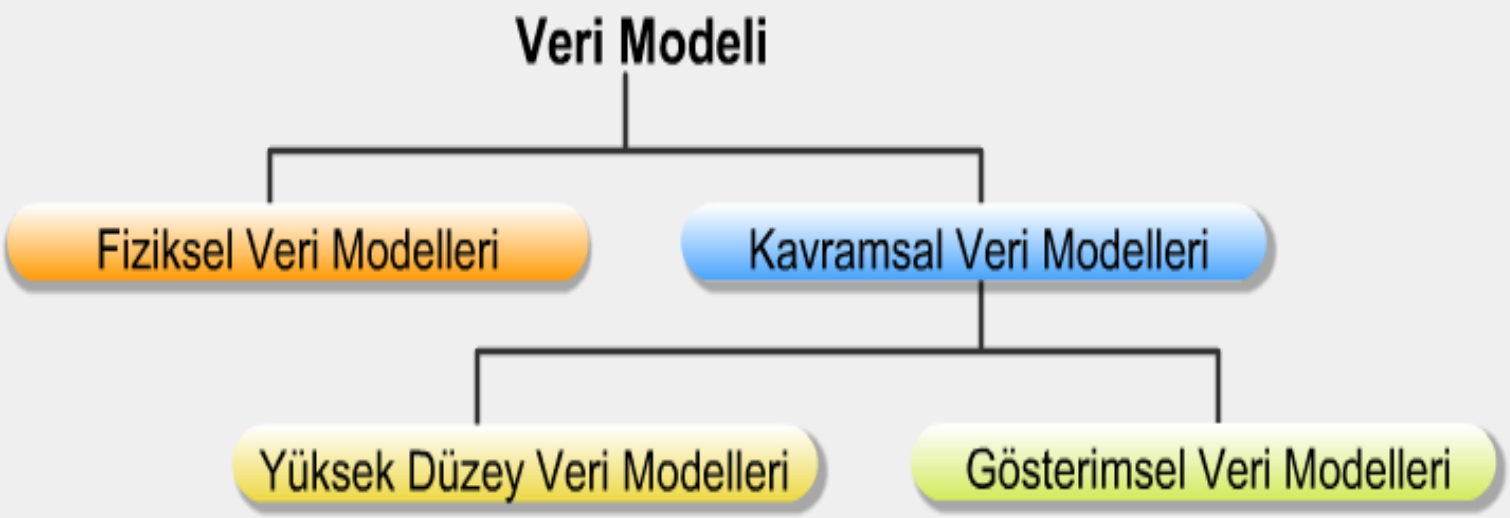
Verinin Birden Fazla İşlem Arasında Paylaşımı

- Çok kullanıcılı VTYS yazılımlarının temel rolü eş zamanlı işlemlerin karışıklık olmadan doğru bir şekilde yapılmasına olanak tanımak işlemlerin doğru olarak yapılmasını garanti etmektir.
- Bu özellik veri tabanı (VT) kavramını, dosya işleme kavramından ayıran en önemli özelliktir. Bu kontrol aynı anda birden çok kullanıcının aynı veriyi güncellemeye çalışmasını, güncellemenin doğru olması açısından garanti eder.

VTYS Kullanımının Ek Yararları

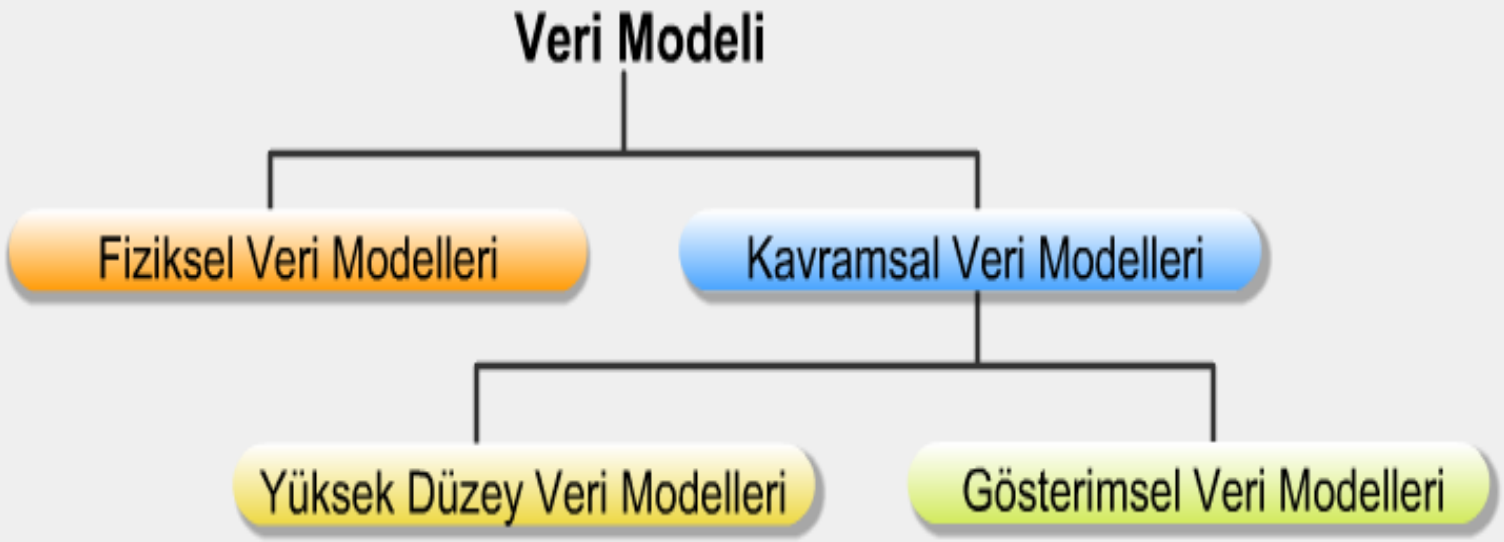
1. Genişleme potansiyeli,
2. Esneklik,
3. Uygulama geliştirme zamanının azalması,
4. Güncel bilgilerin tüm kullanıcılara aynı zamanda ulaşması,
5. Ölçümde ekonomi,
6. İşletme ortamındaki ortak verilerin tekrarının önlenmesi verilerin merkezi denetiminin ve tutarlılığının sağlanması,
7. Fiziksel yapı ve erişim yöntemi karmaşıklıklarının Her kullanıcıya yanlız ilgilendiği verilerin kolay anlaşılır yapılarda sunulması,
8. Uygulama yazılımı geliştirmenin kolaylaşması,
9. Fiziksel yapı ve erişim yöntemi karmaşıklıklarının Her kullanıcıya yanlız ilgilendiği verilerin kolay anlaşılır yapılarda sunulması,

Veri Modelleri



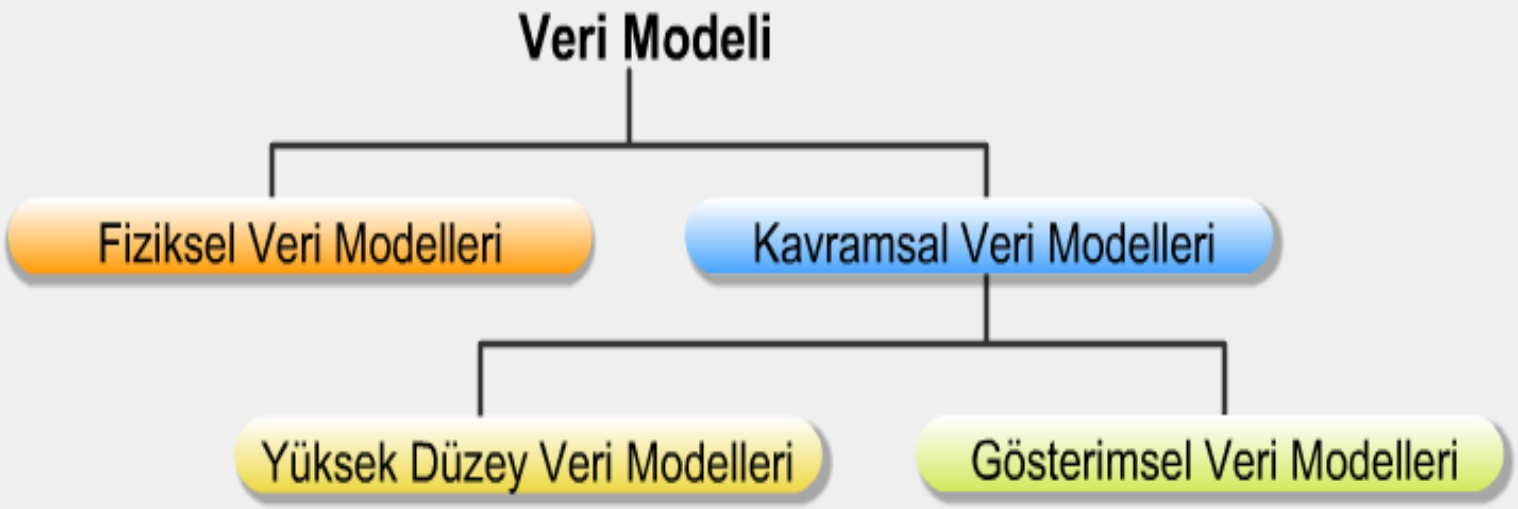
Fiziksel veri modelleri ise verinin bilgisayarda nasıl saklandığının detayları ile ilgili kavramları sağlarlar. Bu ikisinin arasında ise gösterimsel veri modelleri vardır. Bu modeller bilgiyi kayıt biçemi, sırası ve erişim yollarıyla göstererek verinin bilgisayar ortamında nasıl saklandığını açıklar. Burada erişim yolu, belirli bir veritabanı kaydını etkin bir şekilde aramayı sağlayan yapıdır.

Veri Modelleri



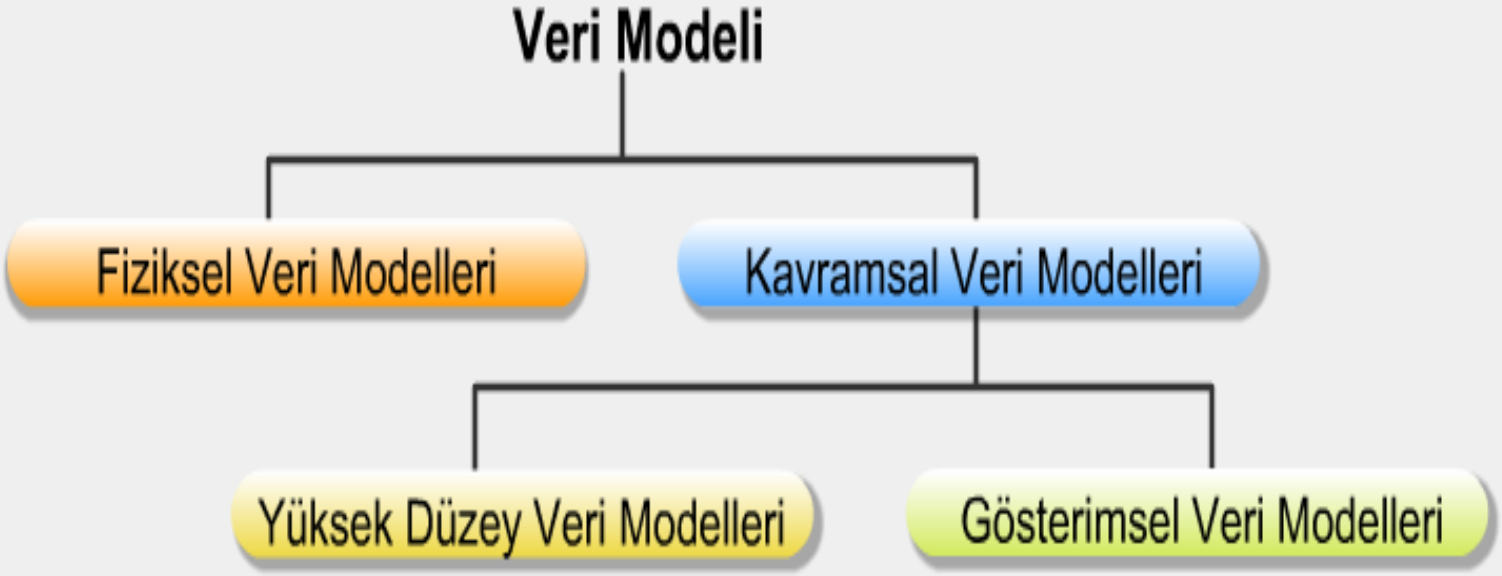
Bunlar hem kullanıcı tarafından anlaşılan hem de verinin bilgisayar içerisindeki gösteriminden çok da fazla uzak olmayan kavramları sağlarlar.

Veri Modelleri



Yüksek düzey veri modelleri varlık, özelik ve ilişki gibi kavramları kullanır. Varlık veri tabanında saklanan ve gerçek dünyadan bir nesne veya kavramdır; proje, işçi gibi. Özelik, varlığı anlatan bir özelliği gösterir işçinin adı, ücreti gibi. Bir veya daha fazla varlık arasındaki ilişki ise bir işçi ve proje arasındaki çalışma ilişkisidir.

Veri Modelleri



Gösterimsel veri modelleri, ticari veri tabanlarında sıklıkla kullanılır ve belli başlı veri üç modeli içerir. İlişkisel, ağ ve hiyerarşik veri modeli. Nesne yönelimli veri modelleri daha yüksek seviyeli gerçekleştirim veri modelleridir ve kavram veri modeline daha yakındır.

Şemalar

- Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önemlidir. Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlandırılır. Veri tabanı şeması, tasarım sırasında belirtilir ve sıkça değişmesi beklenmez.

- Pek çok veri modeli şemaları, diyagramlar halinde göstermek için belli gösterim biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek örneğini göstermez.

VTYS Mimarisi

İçsel Düzey

Veritabanının fiziksel saklanma yapısını açıklar. Fiziksel veri modeli kullanır ve veritabanına erişim yolu ile veri saklamanın tüm detaylarını açıklar.

Kavramsal Düzey

Kavramsal düzey ise kavramsal şema içerir ve kullanıcılar için veritabanının yapısını açıklar. Gerçek fiziksel yapının detaylarını kullanıcıdan gizler, veri tipleri, varlıklar ilişkiler, kullanıcı işlemleri ve sınırlamalar üzerine konsantre olmamızı sağlar. Daha yüksek seviyede veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Dışsal Düzey

Bu düzey, dış şemalar veya kullanıcı görüşleri içerir. Her dış şema veritabanının bir bölümünü açıklar ve her gruba kendi ilgilendiği görüşü sunarken, diğer bir gruptan ilgilenmediği görüşü saklar. Daha yüksek düzeyde veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.

Veri Tabanı Dilleri ve Arabirimleri

Veri Tanımlama Dili (VTD)	Kavramsal şemaları tanımlamak üzere veritabanı yönetici ve tasarımcısı tarafından kullanılır
Saklama Tanımlama Dili (STD)	İçsel şemayı tanımlamak için kullanılır.
Görüş Tanımlama Dili (GTD)	Görüş tanımlama dili kullanıcı görüşlerini tanımlamak ve kavramsal şemaya dönüştürmek amacıyla kullanılır.
Veri İşleme Dili (VİD)	Veri işleme dilidir. Veri tabanı oluşturulduktan sonra Veri tabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır.

Veri Tabanı Sistem Ortamı

- VTYS, karmaşık bir yazılım sistemidir. Burada bir VTYS'yi oluşturan yazılım bileşenlerden söz edilmektedir.
- Tipik bir VTYS yazılımı bileşenleri, VTYS Kataloğu, veri tabanı derleyicisi (VTD), veri yöneticisi, VİD derleyicisi, VT işlemcisi ve yardımcı yazılımlar biçimindedir.

VTYS'nin Sınıflandırılması

- Bir VTYS'ni sınıflandırmada kullanılan temel kriter VTYS'nin dayandığı veri modelidir.

- En fazla kullanılan veri modelleri ilişkisel, ağ, hiyerarşik, nesne-yönelimli ve kavramsal modellerdir.

Hazır Program Kitaplıkları

- Hemen hemen tüm programlama platformlarının kendilerine özgü hazır program kitaplıkları bulunmaktadır.
- Söz konusu kitaplıklar, hemen hemen tüm uygulamalarda kullanılabilecek ortak program parçaları (OCX,VBX vb), sistem yönetimine ilişkin yazılımları içerir.
- Söz konusu yazılımlar program geliştirme hızını oldukça arttırmaktadır. Günümüzde bu tür kitaplıkların edinilmesi, internet sayesinde oldukça kolaylaşmıştır.

CASE Araç ve Ortamları

- Günümüzde bilgisayar destekli yazılım geliştirme ortamları (CASE) oldukça gelişmiş durumdadır.
- CASE araçları, yazılım üretiminin hemen her aşamasında (planlama-çözümleme-tasarım-gerçekleştirim-sınama) üretilen bilgi ya da belgelerin bilgisayar ortamında saklanmasını, bu yolla kolay erişilebilir ve yönetilebilir olmasını olanaklı kılar.
- Hemen tüm CASE araçları, belirli bir standarda uygun olarak geliştirme yapmayı zorlar. Bu yolla yapılan üretimin yüksek kalitede olması sağlanır. CASE araç ve ortamları Bölüm 12'de ayrıntılı olarak incelenmiştir.

Kodlama Stili

- Yazılım Kod stili konusunda herhangi bir kabul edilmiş standart bulunmamaktadır.
- Bu konuda, yazılım geliştiren ekiplerin, kodlama aşamasına başlamadan kodların düzeni konusundaki standartlarını ya da kurallarını geliştirmeleri ve bu kuralların uygulamaya geçmesini sağlamaları önerilmektedir.

Açıklama Satırları

- Bir program parçasını anlaşılabilir kılan en önemli unsurlardan biri bu program kesiminde içerilen açıklama satırlarıdır. Her bir program modülü içerisine

- Modül başlangıç açıklamaları
- Modül Kod Açıklamaları
- Boşluk satırları

eklenmelidir.

Açıklama Satırları

```
/*****************************************************************
*  ©
*****************************************************************
*  Modül Adı:
*  Kütük Adı:
*  İşlevi:
*  Desteklenen Platformlar:
*  Desteklenen Derleyiciler:
*  Taşınabilirlik Konuları:
*  Kullanılan İşlevler:
*  Hatalar:
*  Bilinen Düzeltilmemiş Yanlışlar:
*  Programcı:
*  Tarihçe Günlüğü:
*  İsim                    Tarih                    Açıklama
*
*****************************************************************
```

Kod Biçimlemesi

- Ancak Program kodlama düzeni ise gerçekleştirim aşamasında çözümlenir.
- Programların okunabilirliğini arttırmak, anlaşılabilirliğini kolaylaştırmak amacıyla açıklama satırları kullanımının yanı sıra belirli bir kod yazım düzeni kullanılması gerekmektedir

```
DO I=1 TO N-1;T=I; DO J=I TO I+1;IF A(J)< A(T) THEN DO
T=J;
END;
IF T<>I THEN DO H=A(T);A(T)=A(I); A(I)=T;END;END;
```

(a) Kötü stilde kodlanmış program parçası

```
DO I = 1 TO N-1;
   T=I;
   DO J = 1 TO I+1;
   IF A(J) < A(T) THEN DO
      T=J;
   END;
   IF  T <> I  THEN DO
      H = A(T);
      A(T) = A(I);
      A(I) = H;
   END;
END;
```

(b) İyi stilde kodlanmış program parçası

Anlamlı İsimlendirme

- Hangi değişkenlerin hangi veri tabanı tablosuna ilişkin oldukları,
- Hangi değişkenlerin klavye ve ekran aracılığı ile değer aldıkları
- Hangi değişkenlerin yazıcı çıktılarında görülmek üzere kullanıldıkları,
- Hangi değişkenlerin yalnızca ilgili kod içerisinde ara değişkenler olarak kullanıldıkları

Tekrar

Yapısal Programlama Yapıları

- Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve program karmaşıklığının azaltılması amacıyla "yapısal programlama yapıları" kullanılarak yazılması önemlidir.
- Yapısal Programlama Yapıları, temelde, içinde "go to" deyimi bulunmayan, "tek giriş ve tek çıkışlı" öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır.

Yapısal Programlama Yapıları

Ardışıl İşlem Yapıları

Ardışıl işlemler, herhangi bir koşula bağlı olmaksızın birbiri ardına uyglanması gereken işlemler olarak tanımlanır. Hemen her tür programlama dilinde bulunan, aritmetik işlem deyimleri, okuma/ yazma deyimleri bu tür yapılara örnek olarak verilebilir.

Yapısal Programlama Yapıları

Koşullu İşlem Yapıları

Üç tür Koşullu işlem yapısı bulunmaktadır: tek koşullu işlem yapısı (if-then), iki koşullu işlem yapısı (if-then-else) ve çok koşullu işlem yapısı (case-when). 70'li yılların ortalarından sonra gelen programlama dillerinin hemen hepsinde, bu yapılar doğrudan desteklenmektedir.

Yapısal Programlama Yapıları

Döngü Yapıları

Döngü yapıları, belirli bir koşula bağlı olarak ya da belirli sayıda, bir ya da daha çok kez yinelenecek işlemler için kullanılan yapılardır. Temelde üç tür döngü yapısı bulunmaktadır:

- Belirli sayıda yinelenecek işlemler için kullanılan yapılar (for yapısı),
- Bir koşula bağlı olarak, sıfır ya da birden çok kez yinelenecek işlemler için kullanılan yapılar (while-end yapısı),
- Bir koşula bağlı olarak, bir ya da daha çok kez yinelenecek işlemler için kullanılan yapılar (repeat-until yapısı).

Bu yapıların her biri "tek girişli ve tek çıkışlı" yapılardır. Programlar, yalnızca bu yapılar kullanılarak kodlandığında ve uygun açıklama satırları ile desteklendiğinde, program bakımı kolaylaşmakta ve geliştirme hızlanmaktadır.

PROGRAM KARMAŞIKLIĞI

- Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür.
- Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.

1. Programın Çizge Biçimine Dönüştürülmesi

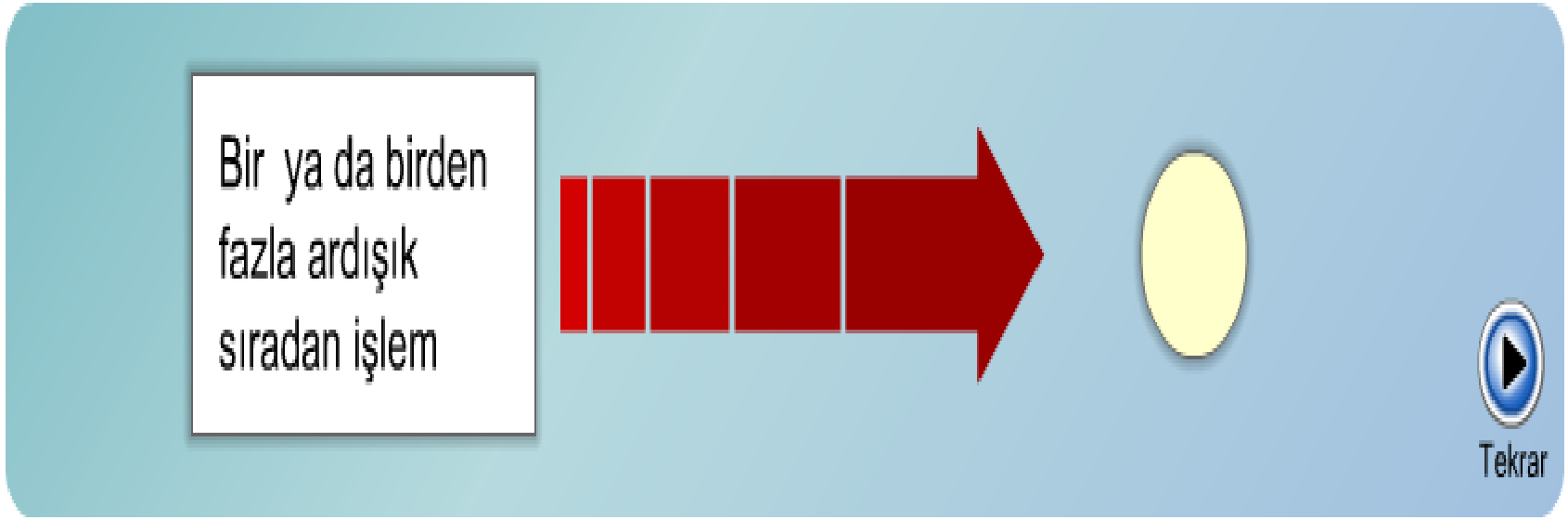
2. Mc Cabe Karmaşıklık Ölçütünün Hesaplanması

Programın Çizge Biçimine Dönüştürülmesi

- Bir program bir ana program ve onunla ilgili birden fazla alt programdan oluşur.
- Gerek ana program gerekse alt programların tümü, McCabe Karmaşıklık ölçütünün hesaplanmasından önce çizge biçimine dönüştürülmelidir.
- Bir program parçasının çizge biçimine dönüştürülmesi için gerekli kurallar aşağıda açıklanmaktadır.

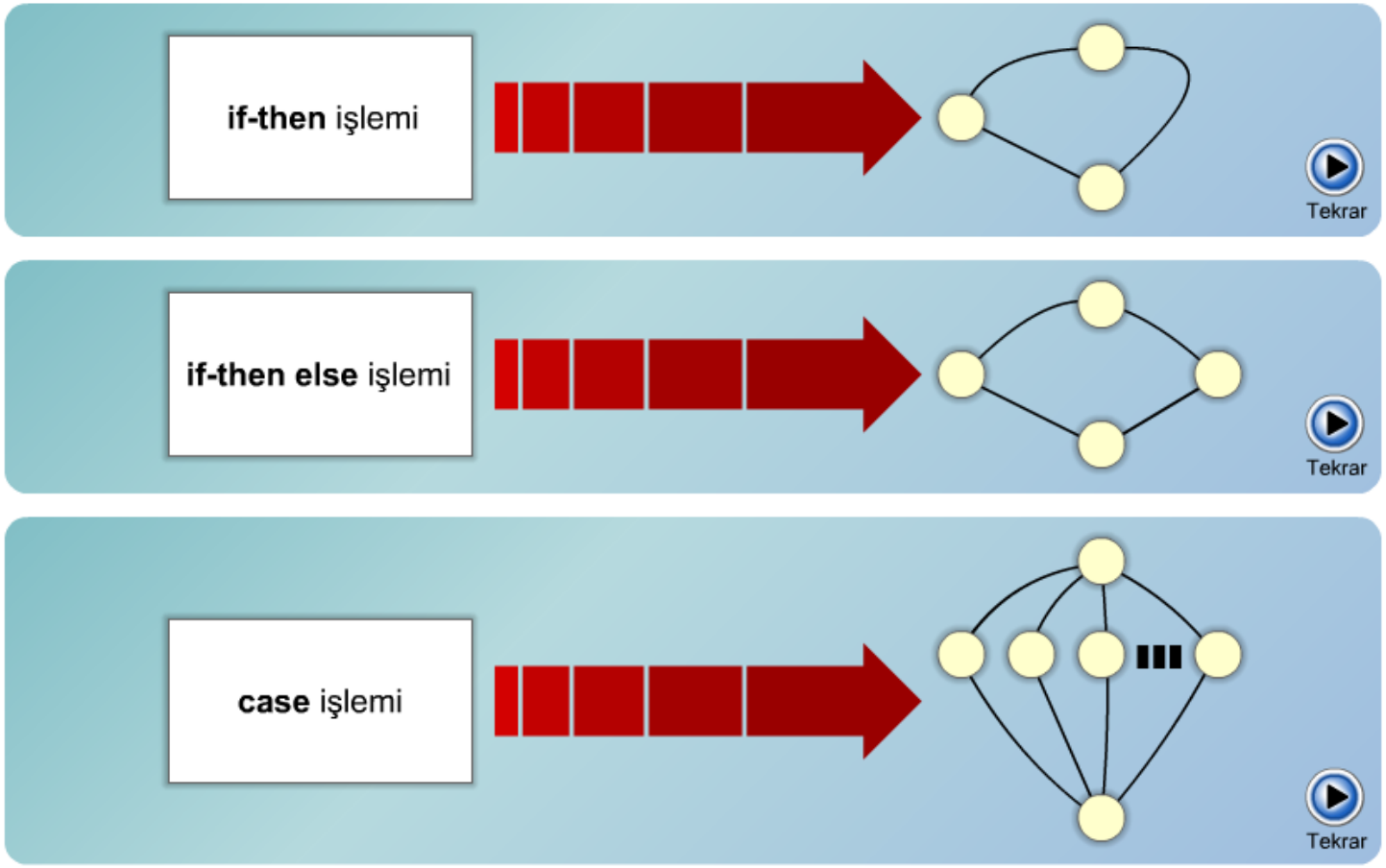
Programın Çizge Biçimine Dönüştürülmesi

1. Sıradan İşlemler: Bir sıradan işlem ya da birden fazla ardışık sıradan işlem çizgede bir düğüme dönüştürülür.



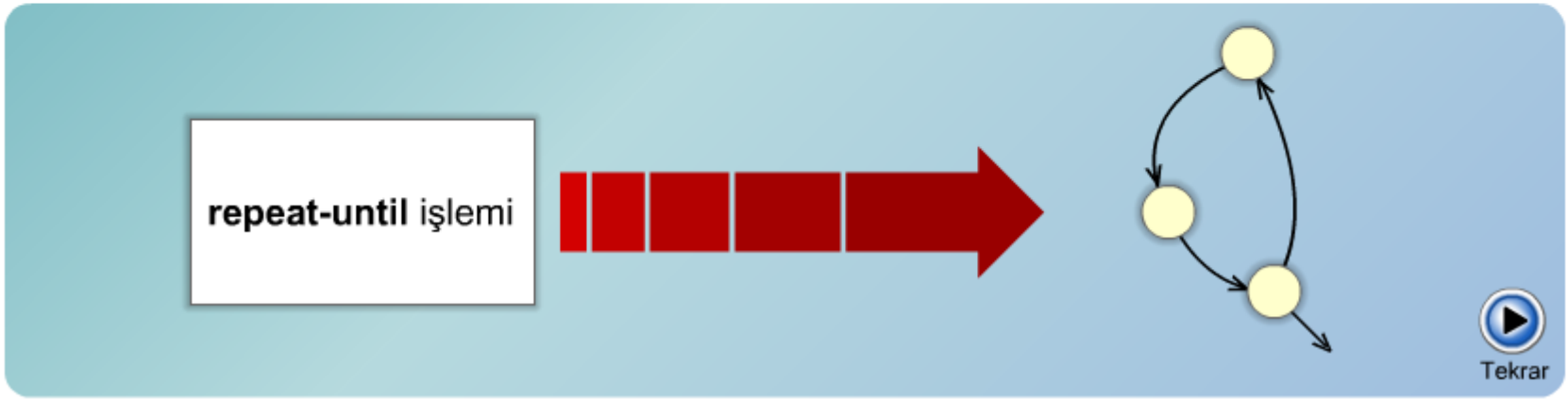
Programın Çizge Biçimine Dönüştürülmesi

2. Koşullu İşlemler:



Programın Çizge Biçimine Dönüştürülmesi

3. Döngü İşlemleri:



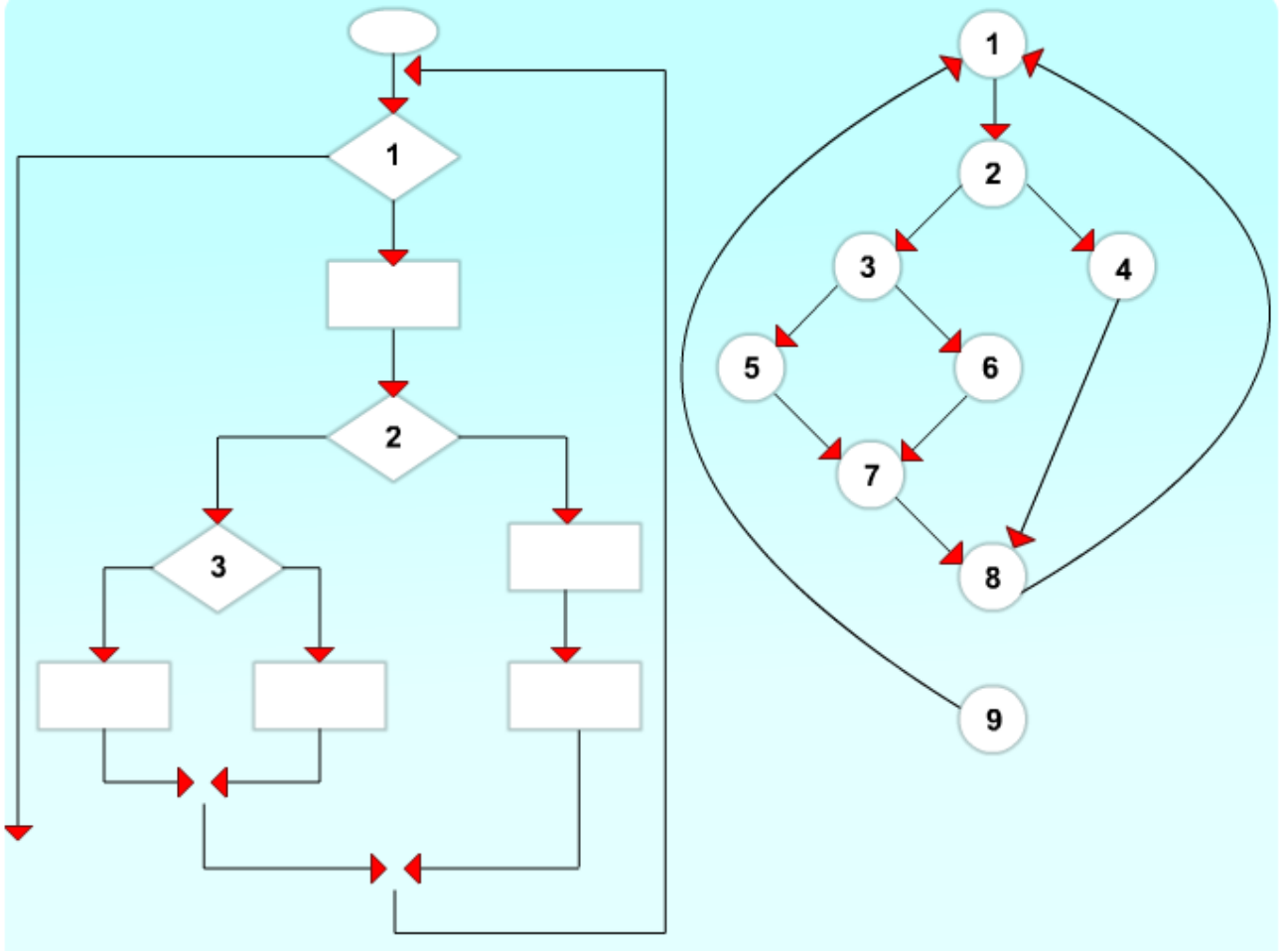
McCabe Karmaşıklık Ölçütü Hesaplama

- Ana program ve alt programlar çizge biçimine dönüştürüldükten sonra programın McCabe karmaşıklık ölçütü (V(G)) aşağıdaki biçimde hesaplanır.

$$V(G) = k - d + 2p$$

- k: kenar sayısı
- d: düğüm sayısı
- p: bileşen sayısı

McCabe Karmaşıklık Ölçütü Hesaplama



McCabe Ölçütü

- Şekil 6.5 (a) da bir programın iş akış şeması verilmekte, (b) kısmında ise önceki kesimde verilen kurallar uygulanarak programın çizge biçimine dönüştürülmüş biçimi verilmektedir. McCabe Karmaşıklığını hesaplamak için kenar ya da dallar ve düğümler sayılıp formülün uygulanması gerekir. Bu örnekte:

- **k = 11** Kenar sayısı
  d = 9 Düğüm sayısı
  p = 1 Bileşen sayısı

karmaşıklık: $V(G) = 11 - 9 + 2 \times 1 = 4$

OLAĞANDIŞI DURUM ÇÖZÜMLEME

- Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.
- Genelde kabul edilen kural, bir programın işletiminin sonlandırılması işleminin -elektrik kesintisi vb donanım hataları dışında- bütünüyle program denetiminde olmasıdır. Bu kural olağan dışı sonlandırma işlemini de kapsamaktadır.
- Bu nedenle program kodları oluşturulurken, olağandışı durumların da dikkate alınması ve bu durumlardaki program davranışına ilişkin yöntemler geliştirilmesi gerekmektedir.

Ada Dili Olağandışı Durum Çözümleme Yapısı

```
begin
     . . . .
     . . . .
     . . . .          Program kod satırları
     . . . .
     . . . .
     . . . .
exception
     when CONSTRAINT_ERROR  =>      SINIR_HATASI;
. .
     when TABLO_DOLU        =>      . . .
end;
```

Olağandışı Durum Tanımları

- Olağan dışı durumlar, programlama dili tarafından tanımlı durumlar olduğu gibi kullanıcı tarafından da tanımlanabilmektedir.

- Yukarıdaki örnekte, CONSTRAINT_ERROR adlı olağan dışı durum ADA Programlama dili tarafından tanımlanmış, TABLO_DOLU adlı olağan dışı durum ise programcı tarafından tanımlanmıştır.

Olağandışı Durum Tanımları

Bu tür olağan dışı durumlar, olağan dışı durumu tanımlayan ve sonucunda "doğru" ya da "yanlış" değeri üreten biri mantıksal yordam ya da fonksiyon tanımından oluşmaktadır.

Şekilde ADA Programlama dili platformunda kendiliğinden tanımlı olağandışı durum örnekleri verilmektedir.

Farklı Olağandışı Durum Çözümleme Yaklaşımları

Anında Durdurma:

Hata bulunduğunda program açıklayıcı bir ileti vererek sona erer. Bu iletinin yanı sıra onaltılı sistemde bir döküm de verir. MS Office paketlerinde alınan "This Program Performed an illegal operation" iletisi ve ardından programın durması, bu tür olağandışı durum çözümleme yaklaşımına örnek olarak verilebilir. Bu yaklaşımda, verilen hata iletisinin anlamlı olması ve programcıya düzeltebilmek amacıyla yol gösterici olması önemlidir.

Hata Kodu Verme:

İlgili program modülünden başarısız şekilde çıkıldığında, programın bir hata kodu vermesi biçminde bir yaklaşımdır. Söz konusu kod "başarılı" ya da "başarısız" durumu belirten mantıksal bir kod olabileceği gibi, bazı uygulamalarda sayısal ya da metin biçiminde olabilir. Bu yaklaşımda, ilgili programında hata oluştuğu anlaşılabilmekte ancak hatanın hangi deyimde oluştuğu anlaşılamamaktadır.

Farklı Olağandışı Durum Çözümleme Yaklaşımları

Tanımlı Olağandışı Yordam Çalıştırma:

Programlama dili platformunun tanımladığı olağan dışı durum çözümleme olanaklarını kullanmak biçimindedir. COBOL Programlama dilindeki "ON READ ERROR" deyimi gibi.

Hata Yordamı Çalıştırma:

Olmayacak Değer Döndürme: Yordamın beklenmeyen bir değer geri döndürmesi gibi bir yaklaşımdır. Örneğin yaş hesaplaması yapıp, yaş bilgisini geri döndüren bir programın eksi değer döndürmesi gibi.

Tekrar

KOD GÖZDEN GEÇİRME

- Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeden okunabilir bir program yazamaz.
- Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işletime alınmamalıdır.
- Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir.

Gözden Geçirme Sürecinin Düzenlenmesi

- Gözden geçirme sürecinin temel özellikleri;
- Hataların bulunması, ancak düzeltilmemesi hedeflenir,
- Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyici kullanılmasıdır.

Gözden Geçirme Sürecinin Düzenlenmesi

- Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.
- Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır.
- Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir. Gözden Geçirme çalışmasının olası çıktıları:

Gözden Geçirme Sürecinin Düzenlenmesi

- Programı olduğu gibi kabul etmek
- Programı bazı değişiklerle kabul etmek
- Programı, önerilen değişikliklerin yapılmasından sonra tekrar gözden geçirmek üzere geri çevirmek.

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.1. Öbek Arayüzü

1. Her öbek tek bir işlevsel amacı yerine getiriyor mu?

2. Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?

3. Öbek tek giriş ve tek çıkışlı mı?

4. Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.2. Giriş Açıklamaları

1. Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?

2. Giriş açıklama satırları, öbeğin amacını açıklıyor mu?

3. Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?

4. Giriş açıklama satırları, çıktıları (parametre, kütük vb) ve hata iletilerini tanımlıyor mu?

5. Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?

6. Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?

7. Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?

8. Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?

9. Her paragrafı açıklayan kısa açıklamalar var mı?

Gözden Geçirme Sırasında Kullanılacak Sorular

6.6.2.3. Veri Kullanımı

1. Işlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?

2. Değişken adları,işlevlerini yansıtacak biçimde anlamlı mı?

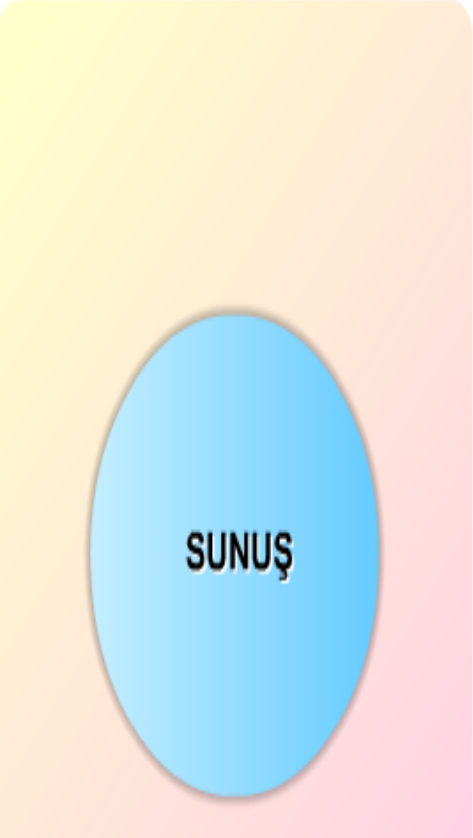
3. Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?

4. Her değişken tek bir amaçla mı kullanılıyor?

5. Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?

6. Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

Gözden Geçirme Sırasında Kullanılacak Sorular



6.6.2.5. Sunuş

1. Her satır, en fazla bir deyim içeriyor mu?

2. Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?

3. Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?

4. Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?

5. Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?

6. Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

Gerçekleştirme Öznitelikleri

- Etkinlik
- Aykırı Durumların Kotarılması
- Hata Ayıklama
- Atık/Çöp toplama
- Soyutlama
- Bilgi Gizleme
- Etiketleme
- Güvenlik
- Basitlik
- Belirgin Arayüz
- Taşınabilirlik
- Çok düzeyli koruma
- Gözden geçirme
- Belgelendirme

Alınan Dersler 1

- Bakım Programcısı kavramı uygulamada çalışmamakta, genellikle yazılımı geliştiren, bakımını da yapmakta ve usta-çırak ilişkisi içinde işler devredilmeye çalışılmaktadır.
- Bu durum, zaten kısıtlı olan bilişim insan gücü kaynağının verimsiz olarak kullanılmasına neden olmakta ve "kişiye bağlı" yazılımlar geliştirilmektedir.

Alınan Dersler 2

- Özellikle C programlama dili için programlama stili çok önemlidir. Sözdizim tanımları gereği C programları, okunmaları ve anlaşılmaları kolay olmayan programlardır.
- Bu nedenle, kodlamaya başlanılmadan önce, bu konuya ilişkin kullanacağınız standartları oluşturulmalı ve kodlama sırasında bu standartlara uyulup uyulmadığını denetlemeniz gerekmektedir.

Alınan Dersler 3

- Açıklama satırları ile ilgili kod uyumlu olmalı. Kod satırları arasında, süreç içerisinde yapılan değişiklikler, yeni eklentiler açıklama satırlarına yansıtılmamakta, bu durum, programların okunabilirliğini giderek zorlaştırmaktadır.

Alınan Dersler 4

- Süreç içinde anlamlı isimlendirmeler kaybolabilmektedir. Program bakımı, belli bir sistematik içerisinde yapılmamakta ve izlenmemektedir.
- Programların, süreç içerisinde, "yamalı bohça" ya dönüşmesi söz konusudur. Bir sistematik ve izleme olmadığında, bakım giderek zorlaşmaktadır.

Alınan Dersler 5

- Gözden Geçirme çalışmasının yapılması amacıyla yönetimi ikna etmek zor.
- Bu tür bir örgütlenmeye sıcak bakılmamakta, öte yandan, programcılar da benzer tepkiyi vermekte ve "sanki kendilerini denetliyorlarmış" mantığı ile davranmaktalar.
- Oysa, Gözden geçirme çalışması yapılmadığında, özellikle büyük yazılım projelerinin geliştirilmesi olanaksız.

Sorular

1. Kendi yazılım geliştirme ortamınızı açıklayınız.

2. Veri tabanı ile veri tabanı yönetim sistemi arasındaki farkı belirtiniz.

3. Veri tabanı Yönetim Sistemi kullanarak, uygulama geliştirme zamanının hasıl kısalacağını açıklayınız.

4. Veri tabanı Yönetim Sistemi kullanımının yararlarını ve aksak yönlerini belirtiniz?

5. Kullandığınız bir veri tabanı yönetim sistemini inceleyiniz. VTD, STD, GİD, GTD dillerinin özelliklerini araştırınız.

6. Kendi kullanımınız için kodlama stili geliştiriniz.

7. Kodlama stilleri ile programlama dilleri arasındaki ilişkiyi belirtiniz.

8. Bildiğiniz bir programlama dilinin, yapısal programlama yapılarından hangilerini doğrudan desteklediğini, hangilerini desteklemediğini araştırınız. Desteklenmeyen yapıların, bu programlama dilinde nasıl gerçekleştirilebileceğini belirtiniz.

9. Verilen bir N doğal sayısının asal olup olmadığını belirleyen bir programı dört değişik biçimde yazınız. Programlar arasındaki zaman farklılıklarını ölçünüz.

10. Program geliştirirken kullandığınız olağan dışı durum çözümleme yöntemlerini açıklayınız?

11. Geliştirdiğiniz bir program için, bölüm içerisinde verilen gözden geçirme sorularını yanıtlayınız. Programınızın niteliği hakkında ne söyleyebilirsiniz?

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

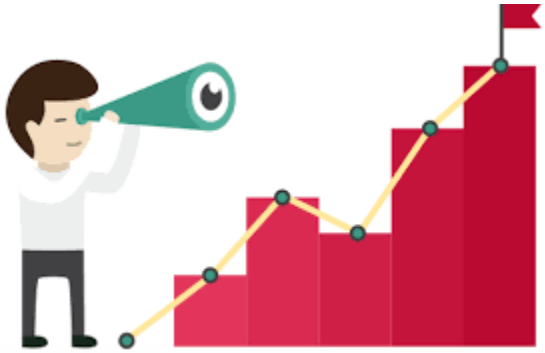
  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ 8.Hafta

Yazılım Doğrulama ve Geçerleme

Bölüm Hedefi

- Yazılım üretimi boyunca, "Doğru Yazılımı mı üretiyoruz?" ve "Yazılımı doğru olarak üretiyor muyuz?" sorularının yanıtlarını araştıran doğrulama ve geçerleme yöntemleri bu bölümde açıklanmaktadır.



Giriş

- Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.

- Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.

Giriş

- Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.
- Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklenmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.

Doğrulama / Geçerleme

Doğrulama

Doğru ürünü mü üretiyoruz?

Ürünü kullanacak kişilerin isteklerinin karşılanıp karşılanmadığına dair etkinliklerden oluşur.

Gerçekleme

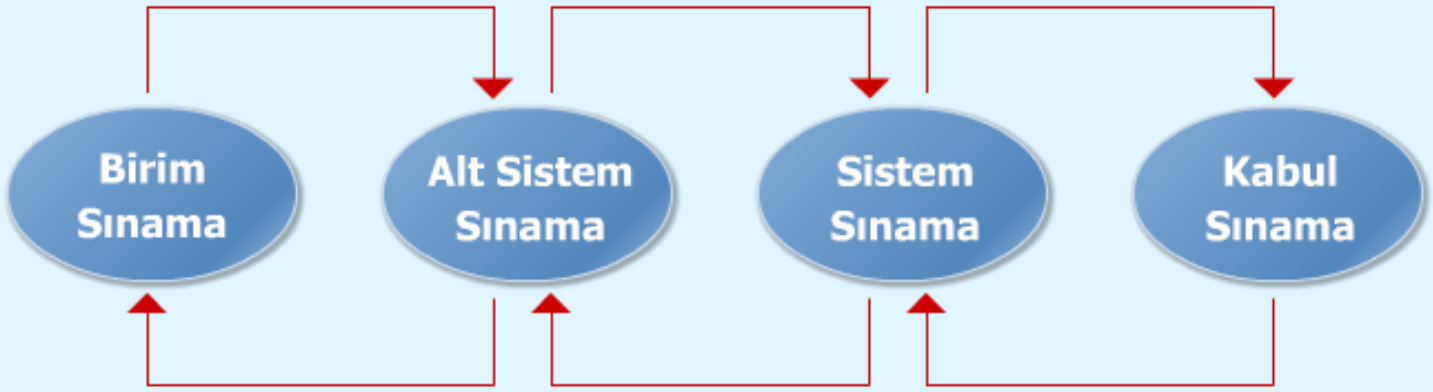
Ürünü doğru olarak mı üretiyoruz?

Ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinden oluşur.

Sınama Kavramları

- Sınama ve bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerin seçimi gerekmektedir.
- Bütünleştirme işleminde, en küçük birimlerden başlanarak sistem düzeyine çıkılmaktadır. Bu değişik düzeylere hitap edecek sınama yöntemleri olmalıdır.

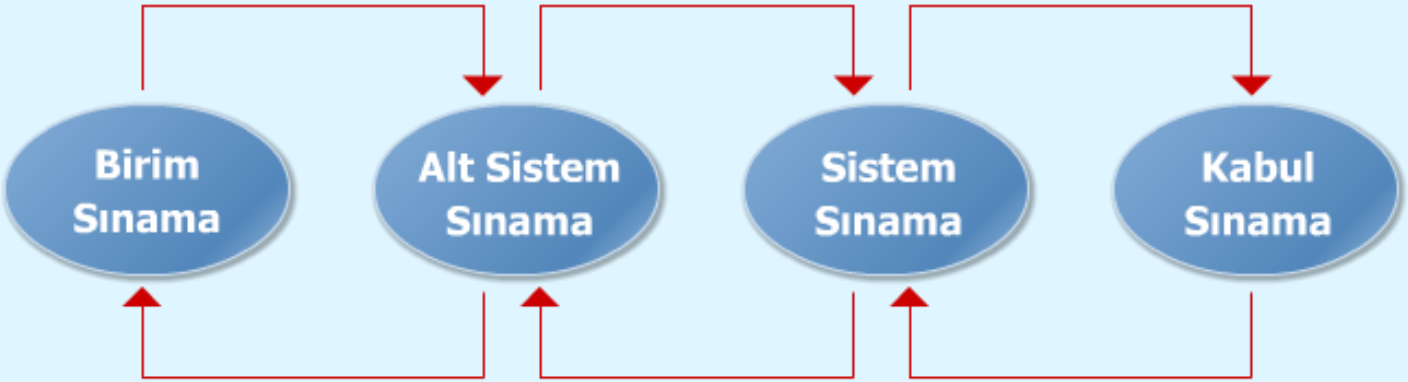
Sınama Kavramları



Birim Sınama

Bağlı oldukları diğer sistem unsurlarından bütünüyle soyutlanmış olarak birimlerin, doğru çalışmalarının belirlenmesi amacıyla yapılır. Birimler, ilişkili yapıtaşlarının bütünleştirilmesinden oluşurlar.

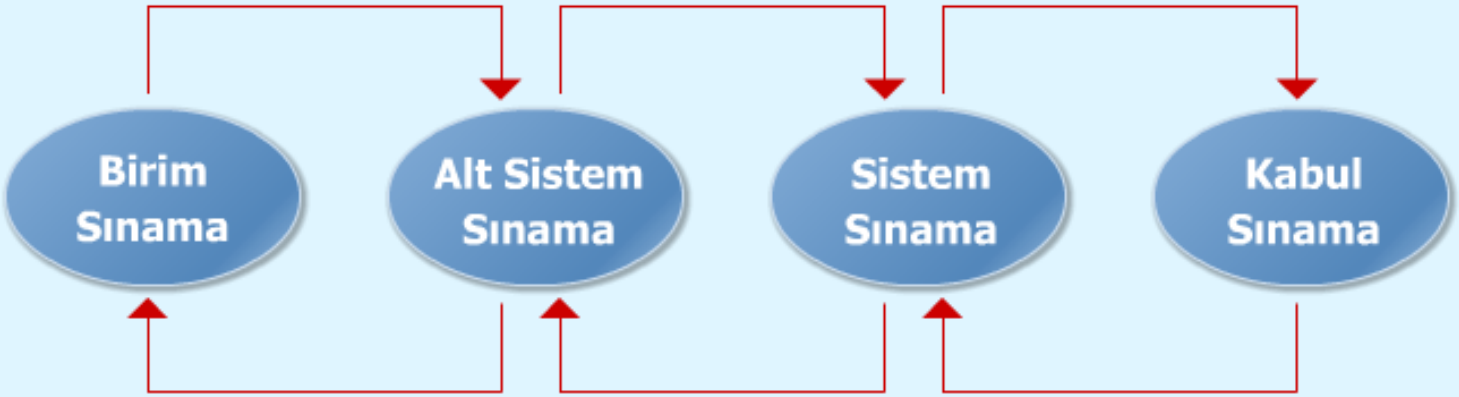
Sınama Kavramları



Alt-Sistem Sınama

Alt sistemler ise, modüllerin bütünleşmesi ile ortaya çıkar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu düzeyde en çok hata arayüzlerde bulunmaktadır, arayüz hatalarına yönelik sınamalara yoğunlaşılmalıdır.

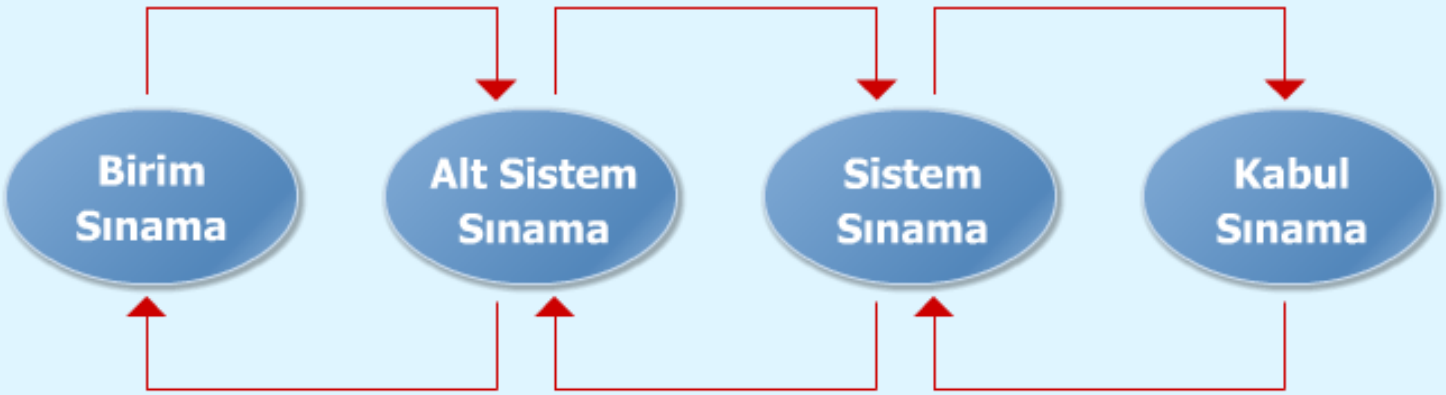
Sınama Kavramları



Sistem Sınaması

Üst düzeyde bileşenlerin sistem ile olan etkileşimlerinde çıkacak hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

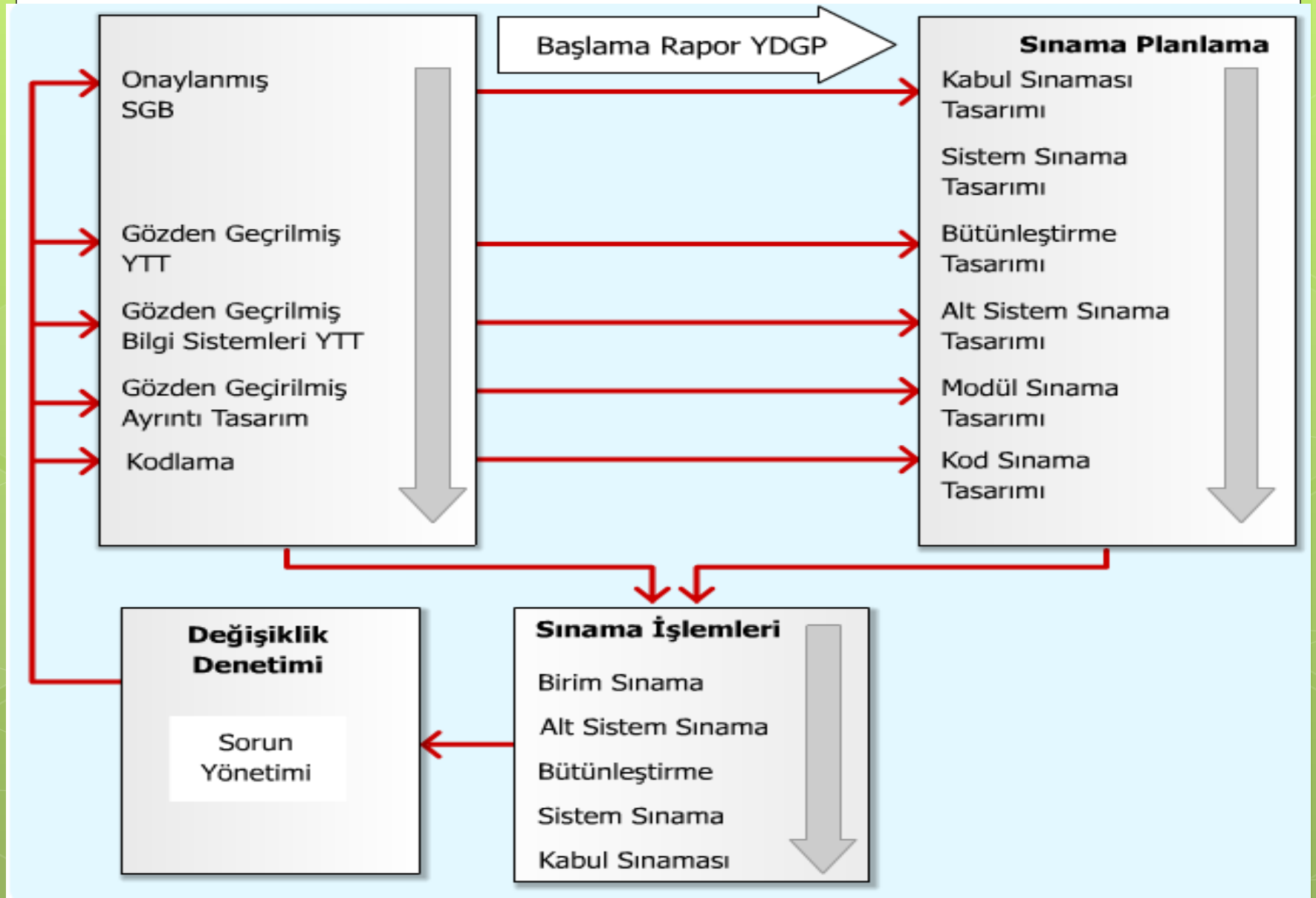
Sınama Kavramları



Kabul Sınaması

Çalıştırılmadan önce sistemin son sınamasıdır. Artık yapay veri yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması ve beta sınaması olarak ta bilinir. Alfa sınamada, tanımında, sınamanın geliştirici organizasyonun yerleşkesinde, kullanıcıların da gelerek katkıda bulunması içerilir. Daha sonra ürünün pazarlama işlemi sırasında beta sınama denilen, sınama, kullanıcının kendi yerleşkesinde, geliştirici gözetiminde yapılır.

Doğrulama ve Geçerleme Yaşam Döngüsü



Sınama Yöntemleri

- Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür.
- Her mühendislik ürünü, iki yoldan biri ile sınanır: Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün **(kara kutu - black box)** veya iç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanması **(beyaz kutu - white box)**.
- Kara kutu sınamasında sisteme, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır. Sonraki sayfalarda beyaz kutu sınamasına ilişkin bilgiler verilmektedir.

Beyaz Kutu Sınaması

- Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır.
- Yapılabilecek denetimler arasında:
  - Bütün bağımsız yolların en azından bir kere sınanması,
  - Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
  - Bütün döngülerin sınır değerlerinde sınanması,
  - İç veri yapılarının denenmesi bulunur.

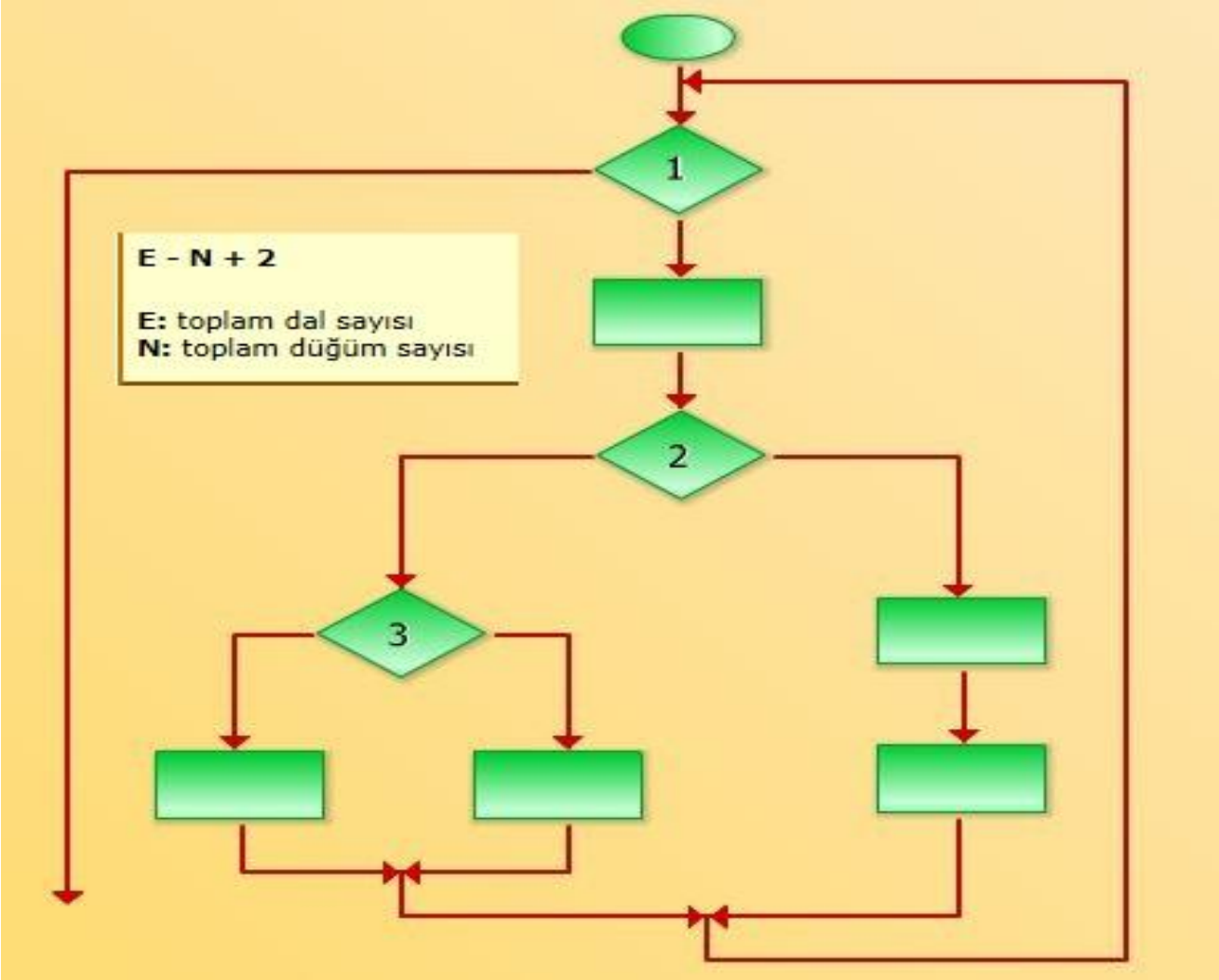
Beyaz Kutu Sınaması

- **Sınamaları yürütürken sınırlı çabamızı yerinde kullanmamız gerekir. Bunun için hataların bazı özelliklerinin bilinmesinde yarar vardır:**
- Bir program kesiminin uygulamada çalıştırılma olasılığı az ise o kesimde hata olması ve bu hatanın önemli olması olasılığı fazladır.
- Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları, aslında çok sıkça çalıştırılıyor olacaktır.
- Yazım hataları rasgele olarak dağılır. Bunlardan bazılarını derleyiciler bulur, bazıları da bulunmadan kalır.

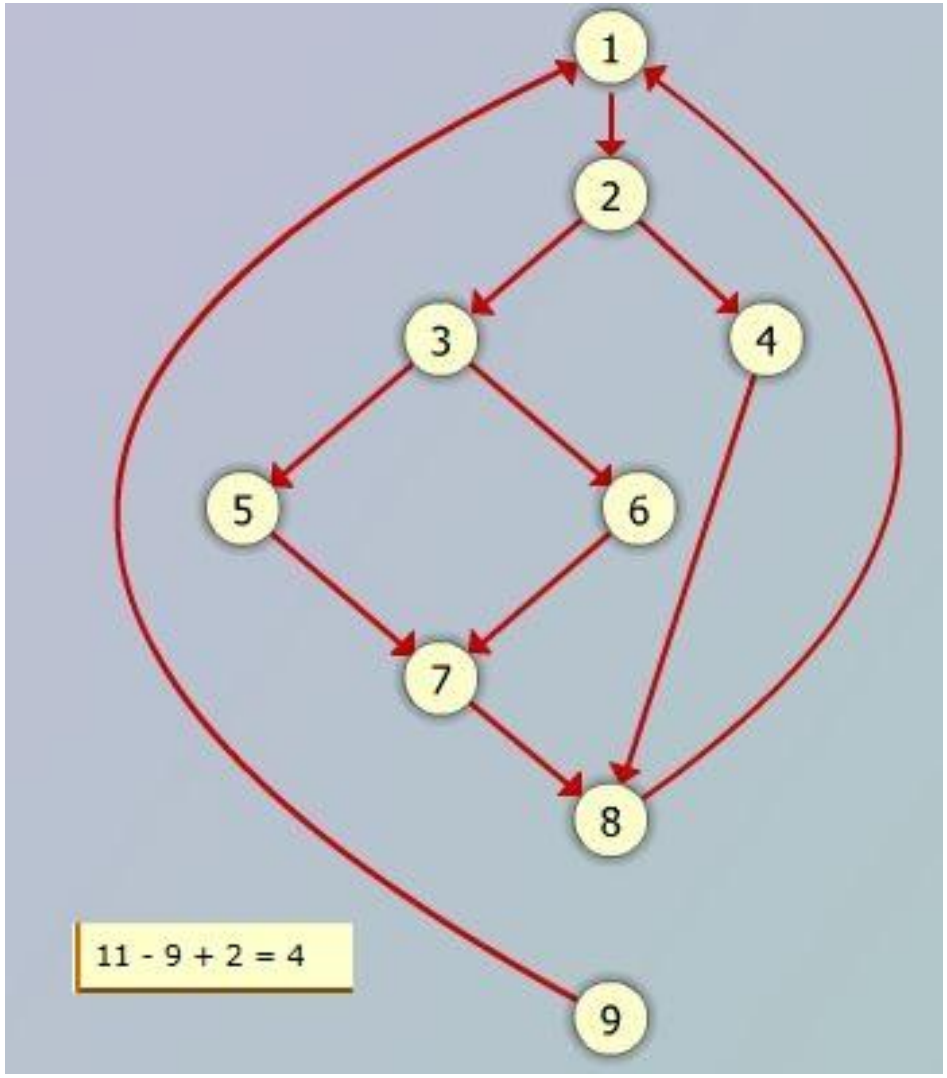
Temel Yollar Sınaması

- Daha önce çevrimsellik karmaşıklığı konusunda gördüğümüz hesap yöntemi ile bir programdaki bağımsız yollar bulunduktan sonra, bu kadar sayıda sınama yaparak programın her birimini bir şekilde sınamalara dahil etmiş oluruz.
- Bağımsız yolların saptanması için önce, program çizgesel bir biçime çevrilir.
- Bunu yapmak için ise, program iş akış şemaları diyagramları iyi bir başlangıç noktasıdır.

Temel Yollar Sınaması



Temel Yollar Sınaması



SINAMA ve BÜTÜNLEŞTİRME STRATEJİLERİ

- Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir.

- Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

Yukarıdan Aşağı Sınama ve Bütünleştirme

- Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur.

- En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir.

Yukarıdan Aşağı Sınama ve Bütünleştirme

- Ancak bu en üstteki bileşenin tam olarak sınanması için alttaki bileşenlerle olan bağlantılarının da çalışması gerekir.

- Alt bileşenler ise bu stratejiye göre henüz hazırlanmış olamazlar. Bunların yerine üst bileşenin sınaması için kullanılmak üzere 'koçan' programları yazılır.

Yukarıdan Aşağı Sınama ve Bütünleştirme

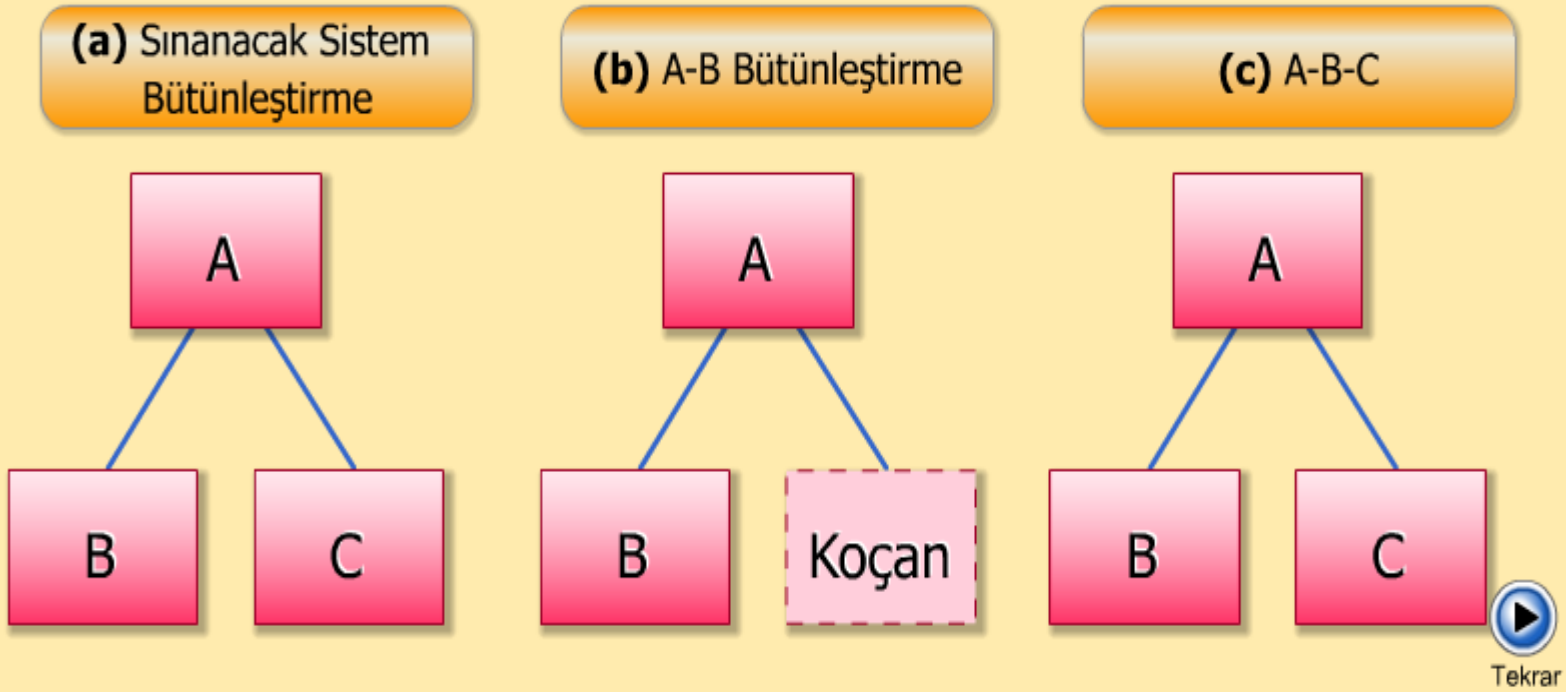
- Koçanlar, bir alt bileşenin, üst bileşen ile arayüzünü temin eden, fakat işlevsel olarak hiç bir şey yapmayan, boş çerçeve programlarıdır.

- Üst bileşenin sınanması bittikten sonra bu koçanlar, içleri doldurularak kendi kodlama ve birim sınama işlemlerini tamamladıktan sonra üst bileşen ile yeniden sınanırlar.

Yukarıdan Aşağı Sınama ve Bütünleştirme

Örnek:

A, B, C birimlerinden oluşan ve birim şeması şekil 7.5(a)'da belirtilen bir sistemin bu tür koçan kullanılarak sınanması şekil 7.5(b) ve şekil 7.5(c)'de belirtilmektedir.



Şekil 7.5: Bütünleştirme Sınamasında "koçan" Kullanımı

İlk adımda A ve B birimleri bütünleştirilir; C için bir "koçan" yazılır. İkinci adımda ise "koçan" kaldırılır ve C ile yer değiştirilerek A-B-C bütünleştirilir.

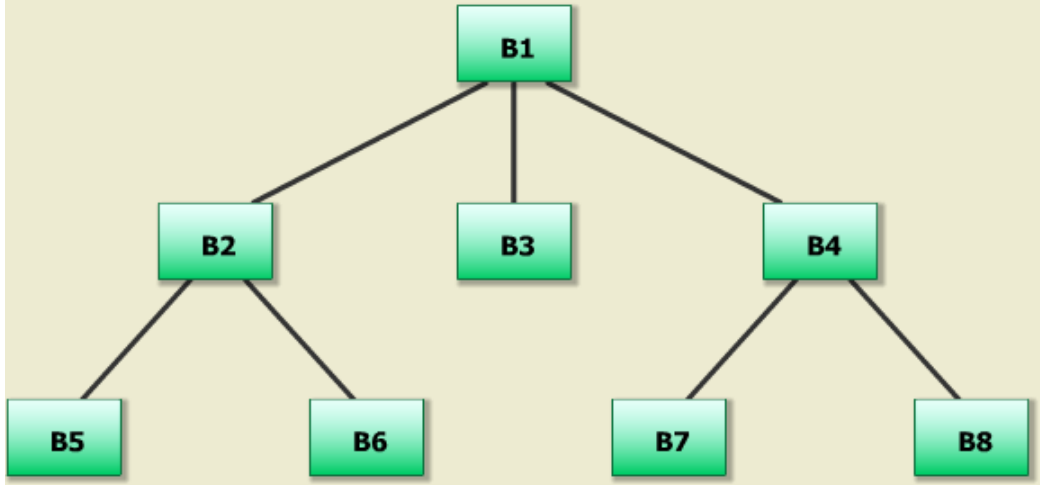
Yukarıdan Aşağı Sınama ve Bütünleştirme

Yukarıdan aşağıya doğru bütünleştirme işleminde iki yaklaşım izlenebilir:

- 1. Yaklaşım: Düzey Öncelikli Bütünleştirme
- 2. Yaklaşım: Derinlik Öncelikli Bütünleştirme

1. Yaklaşım: Düzey Öncelikli Bütünleştirme

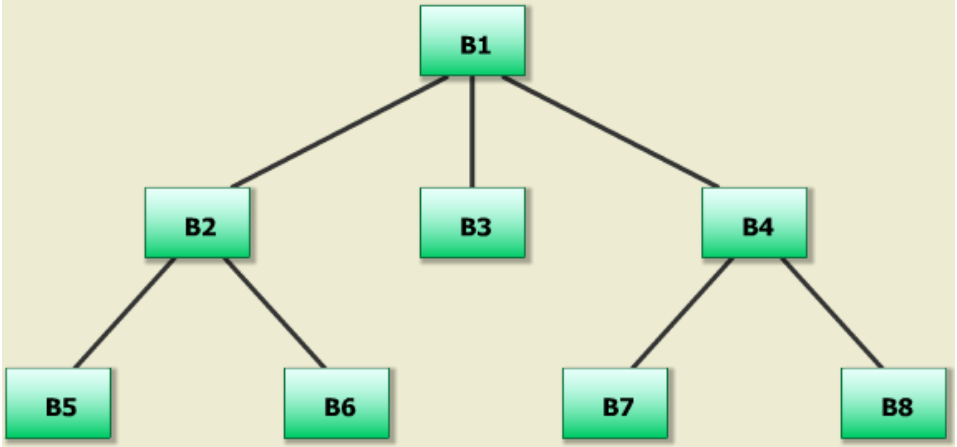
- En üst düzeyden başlanır, öncelikle aynı düzeylerdeki birimler bütünleştirilir.



1. Adım: B1-B2 (KoçanB3 - KoçanB4- KoçanB5 - KoçanB6)
2. Adım: B1-B2-B3 (KoçanB4- KoçanB5 - KoçanB6)
3. Adım: B1-B2-B3-B4 (KoçanB7 - KoçanB8- KoçanB5 - KoçanB6)
4. Adım: B1-B2-B3-B4-B5 (KoçanB7 - KoçanB8- KoçanB6)
5. Adım: B1-B2-B3-B4-B5-B6 (KoçanB7 - KoçanB8)
6. Adım: B1-B2-B3-B4-B5-B6-B7 (KoçanB8)
7. Adım: B1-B2-B3-B4-B5-B6-B7-B8

2. Yaklaşım: Derinlik Öncelikli Bütünleştirme

- En üst düzeyden başlanır. Birim şemasında bulunan her dal soldan sağa olma üzere ele alınır. Bir dala ilişkin bütünleştirme bitirildiğinde diğer dalın bütünleştirmesi başlar.



1. Adım: B1-B2 (KoçanB3 - KoçanB4- KoçanB5 - KoçanB6)
2. Adım: B1-B2 - B5 (KoçanB3- KoçanB4 - KoçanB6)
3. Adım: B1-B2-B5-B6 (KoçanB3 - KoçanB4)
4. Adım: B1-B2-B5-B6-B3 (KoçanB4)
5. Adım: B1-B2-B5-B6-B3-B4 (KoçanB7 - KoçanB8)
6. Adım: B1-B2-B5-B6-B3-B4-B7 (KoçanB8)
7. Adım: B1-B2-B5-B6-B3-B4-B7-B8

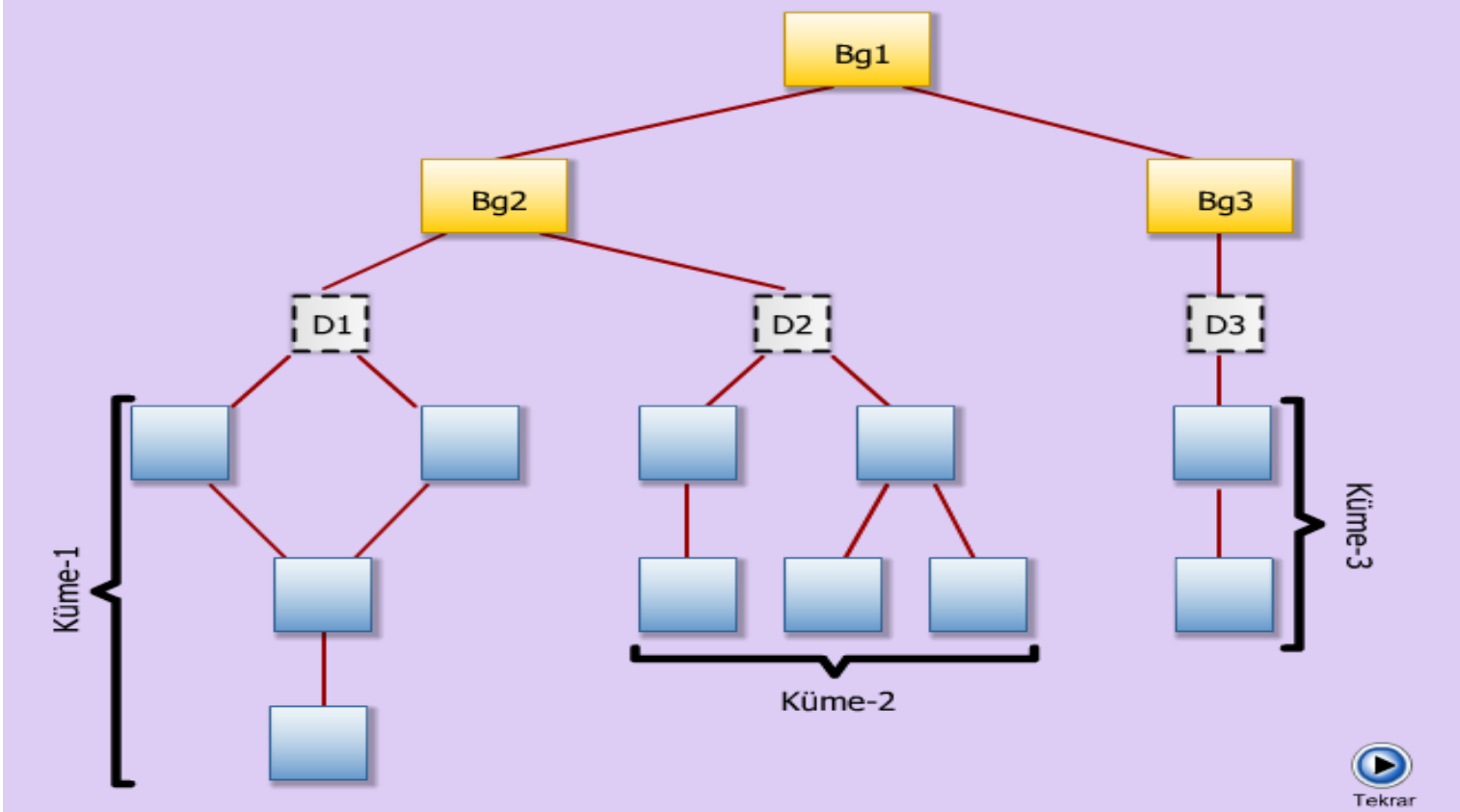
Aşağıdan Yukarıya Sınama ve Bütünleştirme

- Önce en alt düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınanması gerektiğinde bu üst bileşen, bir '**sürücü**' ile temsil edilir.
- Yine amaç, çalışmasa bile arayüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir.

Aşağıdan Yukarıya Sınama ve Bütünleştirme

- Bu kez kodlama, bütünleştirme ve sınama aşağı düzeylerden yukarı düzeylere doğru gelişir ve yukarı düzeylerde önce sürücü olarak yazılan birimler sonra gerçekleriyle yer değiştirerek o düzeyin birimleri/alt sistemleri olurlar.

Aşağıdan Yukarıya Sınama ve Bütünleştirme



Şekil 7.7: Aşağıdan Yukarı Bütünleştirme

Şekilde;

1. Belirli bir yazılım alt işlevini gören alt düzey birimler **kümeler** biçiminde oluşturulurlar,
2. Denetim amaçlı bir **sürücü** programı sınama işlemi için girdi ve çıktı oluşturmak amacıyla yazılır,
3. Sürücüler aşağıdan yukarı kaldırılır ve gerçek birim ya da birim kümeleriyle değiştirilerek sınama işlemi sürdürülür.

SINAMA PLANLAMASI

- Sınama işlemi çok kapsamlıdır. Bir plan güdümünde gerçekleştirilmelidir. Böyle bir planın temel bileşenleri önceki sayfalarda belirtilmiştir.
- Yazılım yaşam döngüsünün süreçlerine koşut olarak, farklı ayrıntı düzeylerinde birden fazla sınama planı hazırlanır.

Giriş
Amaç Tanım ve Kısaltmalar Referanslar

SINAMA PLANLAMASI

Sınama Yönetimi

Sınama Konusu
Sınama Etkinlikleri ve Zamanlama

Temel Sınama Etkinlikleri
Destek Etkinlikler

Kaynaklar ve Sorumluluklar
Personel ve Eğitim Gereksemeleri
Sınama Yaklaşımı
Riskler ve Çözümler
Onaylar

SINAMA PLANLAMASI

Sınama Ayrıntıları

- Sınanacak Sistemler
- Girdiler ve Çıktılar
- Sınamaya Başlanma Koşulları
  - Girdilerin Hazır Olması
  - Ortam Koşulları
  - Kaynak Koşulları
- Sınama Tamamlama Kıstası
- Sınama Geçme-Kalma Kıstası
- Sınama Askıya Alınma Kıstası ve Sürdürme Gerekleri
- Sınama Sonuçları

- Sınama planları; Birim (Modül) Sınama Planı, Alt Sistem Sınama Planları, Bütünleştirme Sınama Planları, Kabul Sınama Planları, Sistem Sınama Planları biçimindedir.

SINAMA BELİRTİMLERİ

- Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir. Bu ayrıtılar temel olarak:
- sınanan program modülü ya da modüllerinin adları,
- sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- sınama verileri,
- sınama senaryoları

türündeki bilgileri içerir.

SINAMA BELİRTİMLERİ

- Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

- Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmasıdır.

SINAMA BELİRTİMLERİ

- Sınama işlemi sonrasında bu belirtimlere,

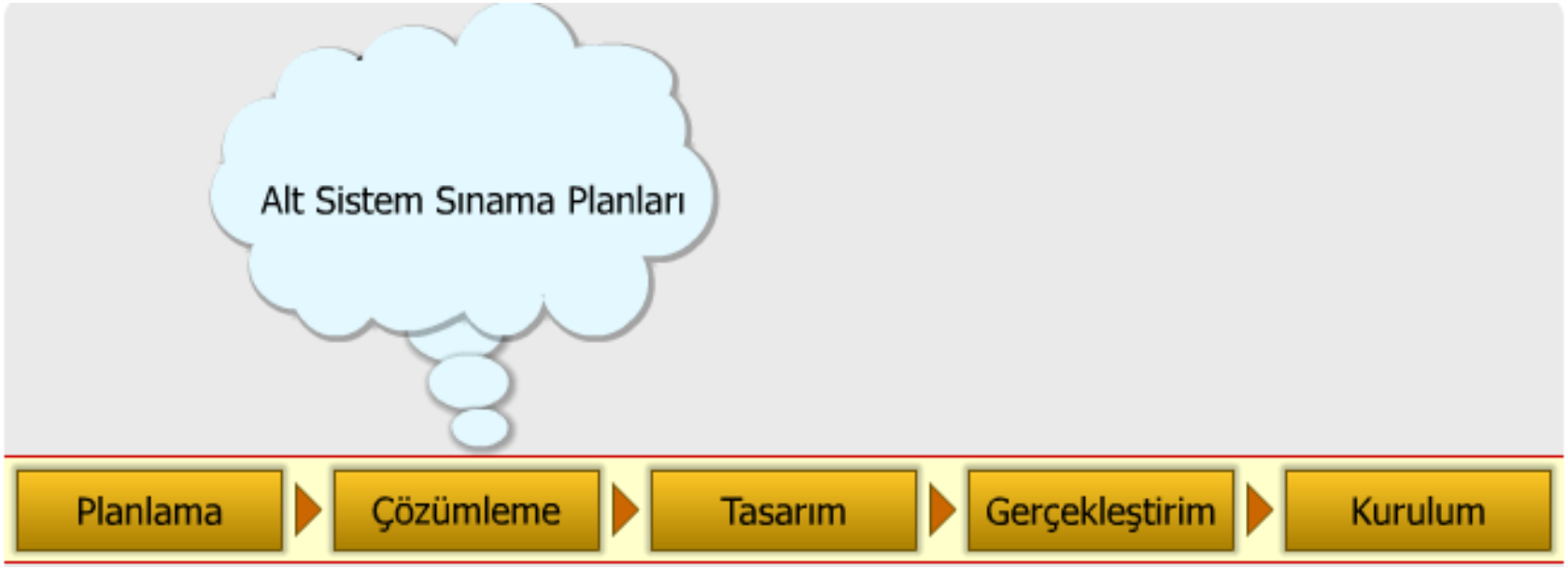
- sınamayı yapan,
- sınama tarihi,
- bulunan hatalar ve açıklamaları

türündeki bilgiler eklenerek sınama raporları oluşturulur.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

Planlama aşamasında genel sınama planı oluşturulur. Söz konusu plan tüm sınama etkinliklerini çok genel hatlarıyla tanımlar ve sınama planlamasında verilen bilgileri içerir.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



Çözümleme aşamasında, sistemler ve alt sistemler ortaya çıkarılır ve sınama planı alt sistemler bazında ayrıntılandırılır.

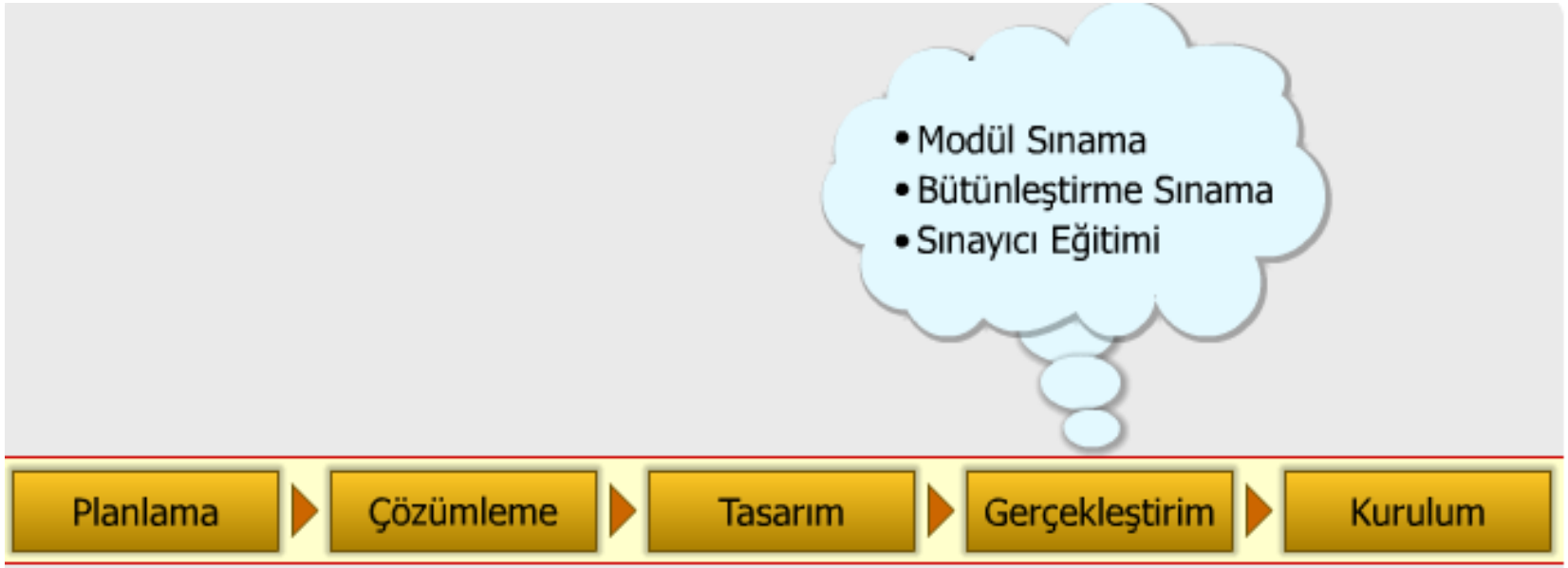
YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- Modül Sınama Planı
- Sınama Belirtimleri
- Sınama Eğitim Kılavuzları

Planlama ▶ Çözümleme ▶ Tasarım ▶ Gerçekleştirim ▶ Kurulum

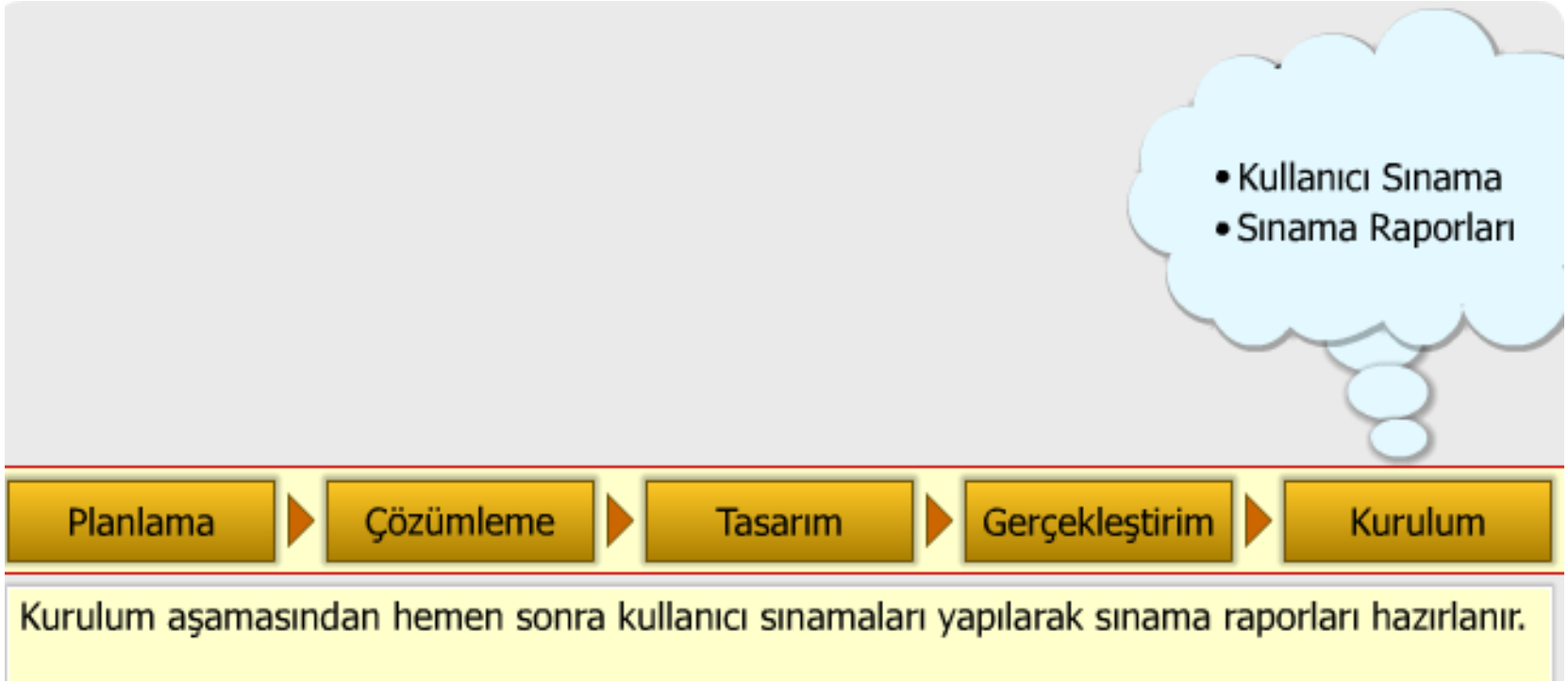
Tasarım aşaması, tüm yazılım modüllerinin ortaya çıkarıldığı aşamadır. Bu aşamanın başlangıcında yazılım modülleri için sınama planı detaylandırılır ve sınama belirtimleri hazırlanır. Söz konusu belirtimler, kullanıcı kılavuzları ile birlikte sınamaya ilişkin eğitim için temel bilgileri oluşturur.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



Gerçekleştirim aşamasında teknik sınamalar yapılır, sınama raporları hazırlanır ve kullanıcı sınayıcıları eğitilir.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ



YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- Sınama sırasında bulunan her hata için, değişiklik kontrol sistemine (DKS), "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

Onulmaz Hatalar: BT projesinin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

- **Küçük Hatalar:** Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.

- **Şekilsel Hatalar:** Heceleme hatası gibi önemsiz hatalardır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Bu kısımda, gerçek yaşam ortamında, **Oracle Designer CASE** aracı ve **Developer** görsel yazılım geliştirme platformu kullanılarak geliştirilen yazılım modüllerinin sınanması işleminin nasıl yapılacağı ve buraya kadar açıklanan sınama yöntemlerinin nasıl uygulandıkları bir örnek üzerinde anlatılmaktadır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Oracle Developer kullanılarak geliştirilen her yazılım **form**lardan oluşur.
- Bir form, bir ekran ve bu ekranda yapılan işlemlere karşılık gelen PL/SQL kodları biçiminde tanımlanır.
- Bu örnekte elimizde, sınama işlemine koşulacak ve uygulamanın çeşitli işlevlerine ilişkin bir dizi form olduğunu düşünebiliriz.

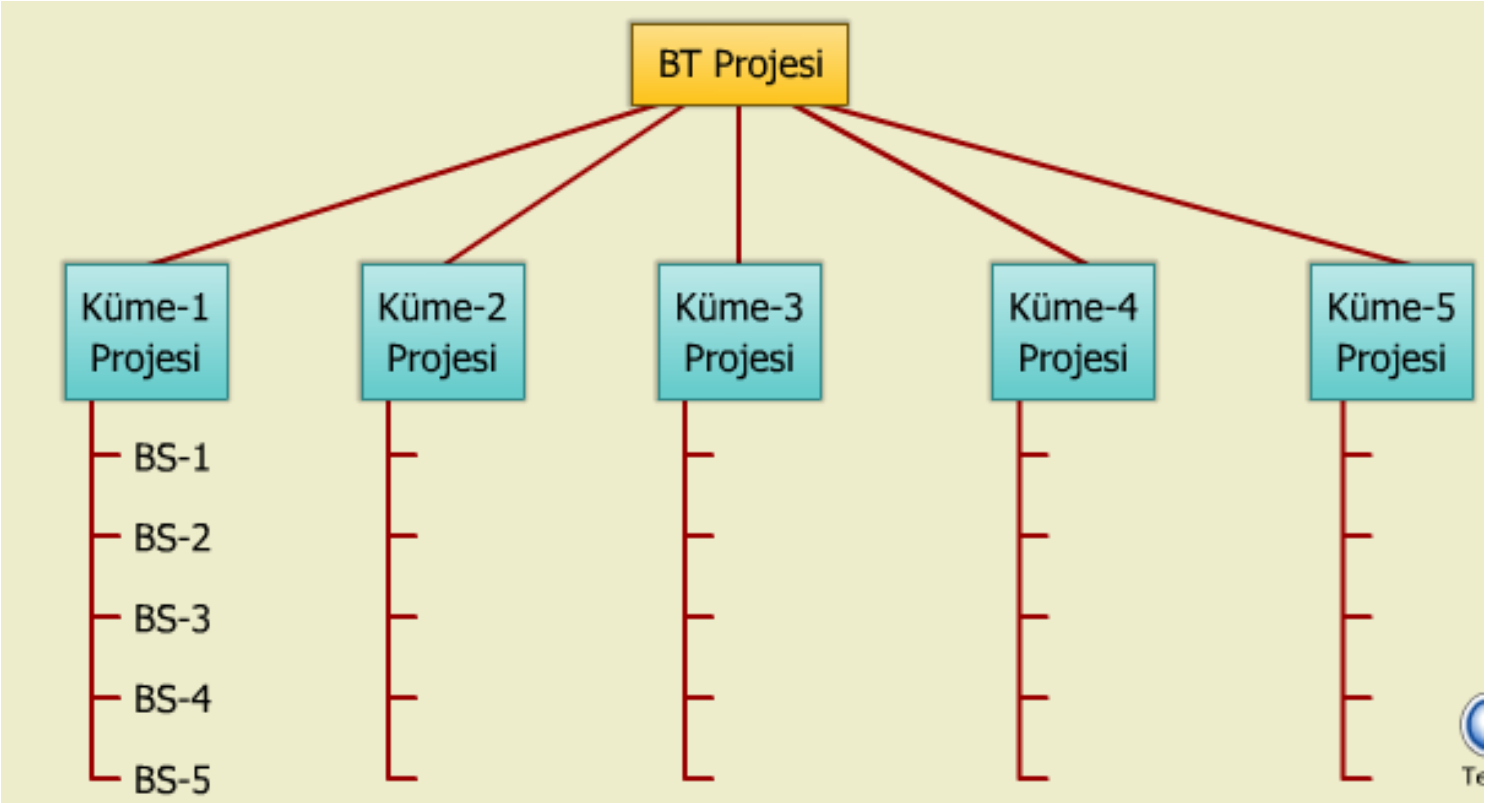
BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Bu örnekte söz edilen uygulama, 2000'den fazla kullanıcısı olan, ülkenin çeşitli yörelerine dağılmış birimlerde çalışacak biçimde tasarlanmış ve 1000'den fazla Developer formundan oluşmaktadır.

BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama

- Uygulamanın sınama aşamasına gelmesi, 2 yıllık bir süre ve yaklaşık 100 kişi-yıl'lık bir iş gücü gerektirmiştir.
- Uygulama beş ana kümeye bölünmüş ve her küme belirli sayıda bilgi sistemini içermektedir.
- Toplam olarak 30 bilgi sistemi bulunmaktadır. Uygulama sıra düzeni Şekil 7.10'da verilmektedir.

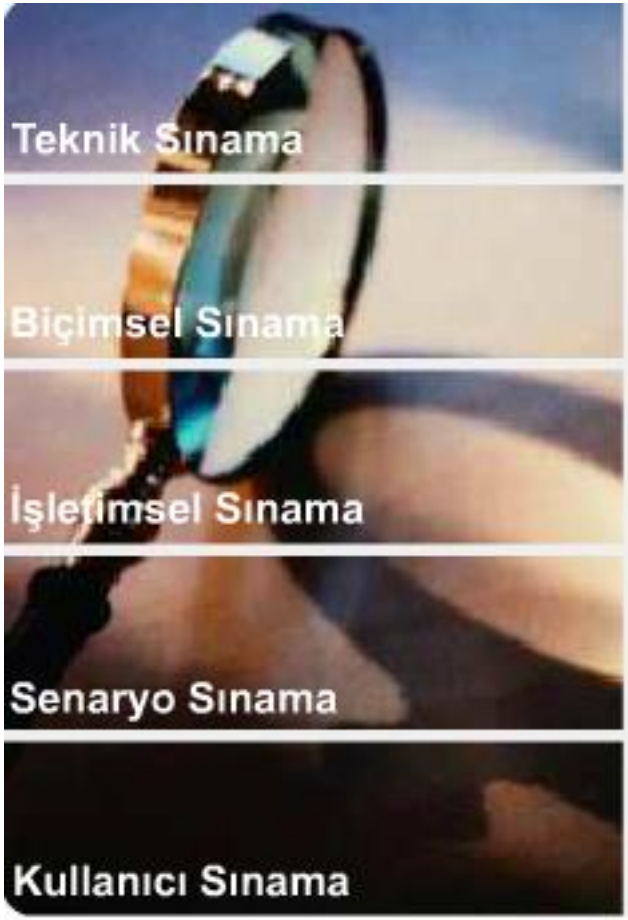
BİR UYGULAMA: Görsel Yazılım Geliştirme Ortamında Sınama



Sınama Ortamı Oluşturulması

- Üretimin etkilenmemesi amacıyla, yalnızca sınayıcıların kullanacakları ve ayrı bilgisayarlardan oluşan bir sınama ortamı oluşturuldu.
- Oluşturulan sınama ortamı ile üretim ortamının birebir aynı olması sağlandı.
- Üretimi biten yazılım parçaları, bir kayıt düzeni içerisinde sınama ortamına alındı.

Sınama Yöntemlerine Karar Verilmesi

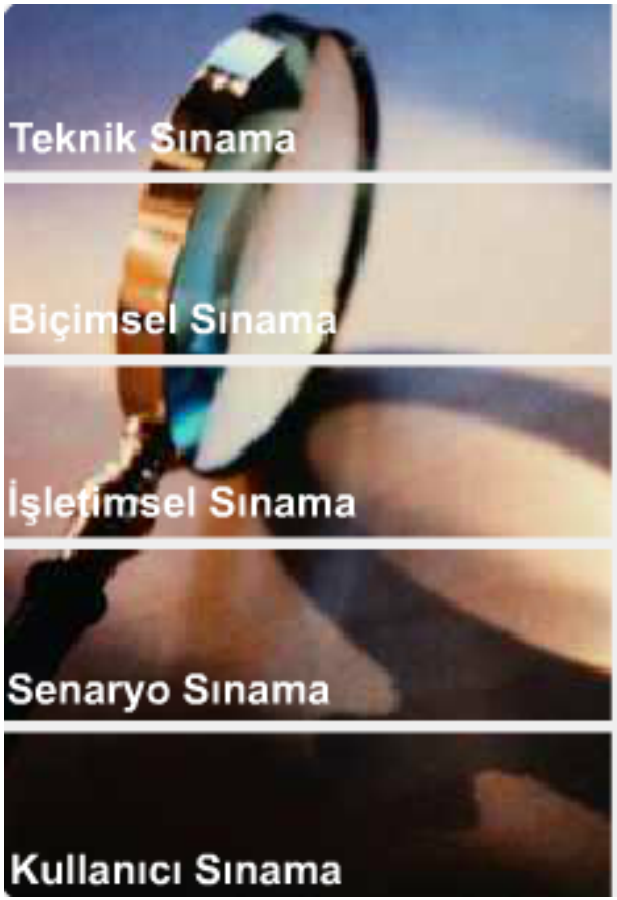


Teknik Sınama

Üretim ortamında yapılacak sınama olarak karar verildi. Bu sınama, modül sınaması ve bütünleştirme sınaması olarak üretim ekipleri tarafından gerçekleştirildi. Modül sınama yöntemi olarak "beyaz kutu" sınama yöntemi uygulandı. Tüm program deyimleri en az bir kez, tüm döngüler en az 10 kez yinelenecek biçimde sınama yapıldı.

Bütünleştirme sınama yöntemi olarak, "yukarıdan aşağıya sınama yöntemi" ve "derinlik öncelikli bütünleştirme" stratejisi uygulandı.

Sınama Yöntemlerine Karar Verilmesi



Biçimsel Sınama

Üretim ekiplerinden bağımsız olarak, sınama ekipleri tarafından yapılan sınamadır. Bu sınama, Developer formları üzerinde görsel olarak yapıldı. Amaç, formların, önceden kararlaştırılan standartlara uygunluğunun saptanmasıydı. Örneğin, form alanları, kararlaştırılan uzunlukta mı? Başlıklar istenilen gibi koyu mu? Yardım düğmesi hep aynı yerde mi vb.

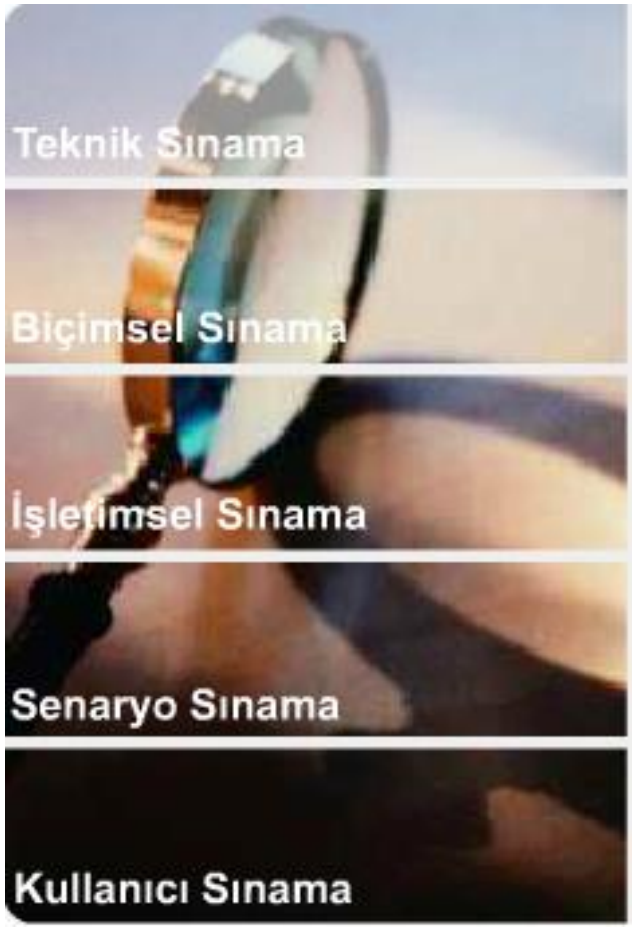
Sınama, formlar işletilmeden yapılır. Tüm formlar tek tek incelenir ve standartlara uygun olmayanlar belirlenip, düzeltilmek üzere üretim ekibine geri iletilir. Biçimsel sınamaların yapılması amacıyla denetim listeleri hazırlanır ve sınama sırasında bu listeler kullanılır. Listelere kaydedilen her sonuç DKS'ye aktarılır. Bu yolla üretim ekiplerinin performansı izlenebilir.

Sınama Yöntemlerine Karar Verilmesi

İşletimsel Sınama

Üretim ekiplerinden bağımsız olarak, sınama ekipleri tarafından yapılan sınamadır. Biçimsel sınama işlemi bittikten sonra yapılır. Bu sınamada her form ayrı ayrı çalıştırılarak işlem yapılır. Amaç, formun çalışıp çalışmadığının belirlenmesidir. Form alanlarının sınır değerlerle çalışıp çalışmadığı, aykırı değer verildiğinde uygun hata iletisi alınıp alınmadığı vb belirlenmeye çalışılır.

Sınama Yöntemlerine Karar Verilmesi

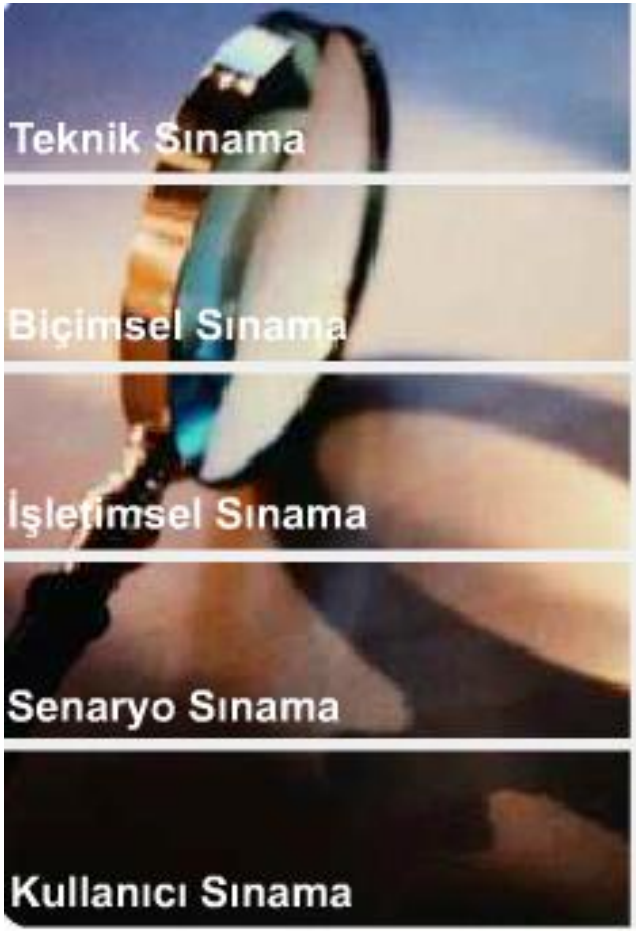


Senaryo Sınaması

Sınama ekipleri tarafından yapılan sınamadır. Ancak, senaryoların hazırlanması sırasında üretim ekipleri ile birlikte çalışılır.

Amaç, birden fazla formun bir arada sınanmasıdır. Bu amaçla, “senaryo”lar hazırlanır. Her senaryo, çözümleme aşamasında belirlenen bir iş fonksiyonuna karşılık gelecek biçimde. hazırlanır.

Sınama Yöntemlerine Karar Verilmesi



Kullanıcı Sınaması

Kullanıcılar tarafından yapılması öngörülen sınamadır. Senaryo sınamasının kullanıcı tarafında yapılan biçimi olarak düşünülebilir.

Kullanıcı Sınama Eğitimi

- Sınama yapılacak kullanıcı sınayıcılarına, sınamaların nasıl yapılacağına ilişkin eğitim verilmesi gerekmektedir.
- Eğitim kitapçıklarının hazırlanması amacıyla, senaryo sınamalarında kullanılan "senaryo"lar ve kullanıcı kitapçıkları kullanılır.

Sınamaların Yapılması

Sınamalar sırasıyla:

- Teknik Sınama,
- Biçimsel Sınama,
- İşletimsel Sınama,
- Senaryo Sınaması,
- Kullanıcı Sınama

biçiminde yapılır.

Alınan Dersler 1

- Bir taraftan üretimin yapıldığı, öte yandan ise kullanıcı tarafında sınamaların yapıldığı bir ortamda, üretim ekipleri bir yandan yeni yazılım parçaları geliştirme, öte yandan ise sınama sonucu bildirilen hataları düzeltme durumu ile karşı karşıyadır.

Alınan Dersler 1

- Bu durum, zaman zaman üretim ekiplerinde dirençlere neden olmaktadır.
- Büyük projeler için, kaçınılmaz olan bu tür durumların iyi izlenmesi ve planlanması gerekmektedir.
- Bu anlamda Kalite ekibi oldukça önem kazanmaktadır.

Alınan Dersler 2

- Sınama yapılacak ortam ile üretim ortamının, fiziksel olarak birbirinden ayrı olarak düzenlenmesi çok önemidir.
- Aksi durumda, sınama sırasında, sistemlerin kilitlenmesi ve veri tabanının zarar görmesi vb. sorunlarla karşılaşılır.
- Yapılan işlerin izlenmesi zorlaşır.

Alınan Dersler 3

- Kullanıcı sınayıcı eğitimlerinin zaman zaman yinelenmesi gerekir.

- Kurumlardaki eleman değişiminin fazla olması sonucu, eğitim almamış kullanıcıların sistemi sınaması gibi durumlarla karşılaşılır ki bu da projeyi olumsuz olarak etkiler.

Alınan Dersler 4

- Kullanıcı sınayıcıları, kendi işlerinin yoğunluğunu öne sürerek, sınama işlemine gereken önemi gösterememektedir.
- Bu durumda Yerinde Destek ekiplerine önemli görevler düşmektedir.

Sorular

1. Doğrulama ile Geçerleme arasındaki farklılıkları belirtiniz. Birer örnekle açıklayınız.

2. Sınama Yöntemlerini açıklayınız.

3. "Beyaz Kutu" sınama ile "Temel Yollar Sınama" yöntemleri arasındaki farlılıkları belirtiniz.

4. Sınama Yöntemleri ile sınama belirtimleri arasındaki farkı belirtiniz.

5. Yukarıdan aşağıya doğru bütünleştirme ve aşağıdan yukarıya bütünleştirme yöntemlerinin zorluklarını ve kolaylıklarını belirtiniz.

6. Sınama belirtimlerinin önemi nedir.

7. Kullanıcı sınaması sırasında yaşanabilecek sorunları belirtiniz.

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

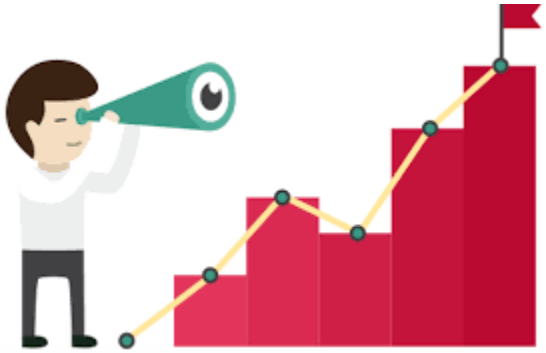
  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

YAZILIM MÜHENDİSLİĞİNİN TEMELLERİ
9.Hafta

Bakım

Bölüm Hedefi

- Geliştirilen yazılımın uygulamaya alınabilmesi için gerekli yöntemler ve yazılımın çalışması sırasında yapılması gereken bakım işlemleri bu bölümde açıklanmaktadır. Bölümün temel hedefleri arasında, IEEE 1219 standardının tanıtımı bulunmaktadır.



Giriş

- Sınama işlemleri bitirilen yazılımın kullanıcı alanına yüklenmesi ve uygulamanın başlatılması gerekmektedir.
- Yazılım kullanıma geçtikten sonra, yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan "bakım" aşaması başlar.
- İzleyen kesimlerde, kurulum ve bakım aşamasında yapılması gerekenler açıklanmaktadır.
- Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı dikkate alınmıştır.

KURULUM

- Sınanmış yazılımların kullanıcı sahasına aktarılması ve yazılımın gerçek yaşamda uygulamasının başlatılması için yapılan işlemler kurulum işlemleri olarak tanımlanmaktadır.

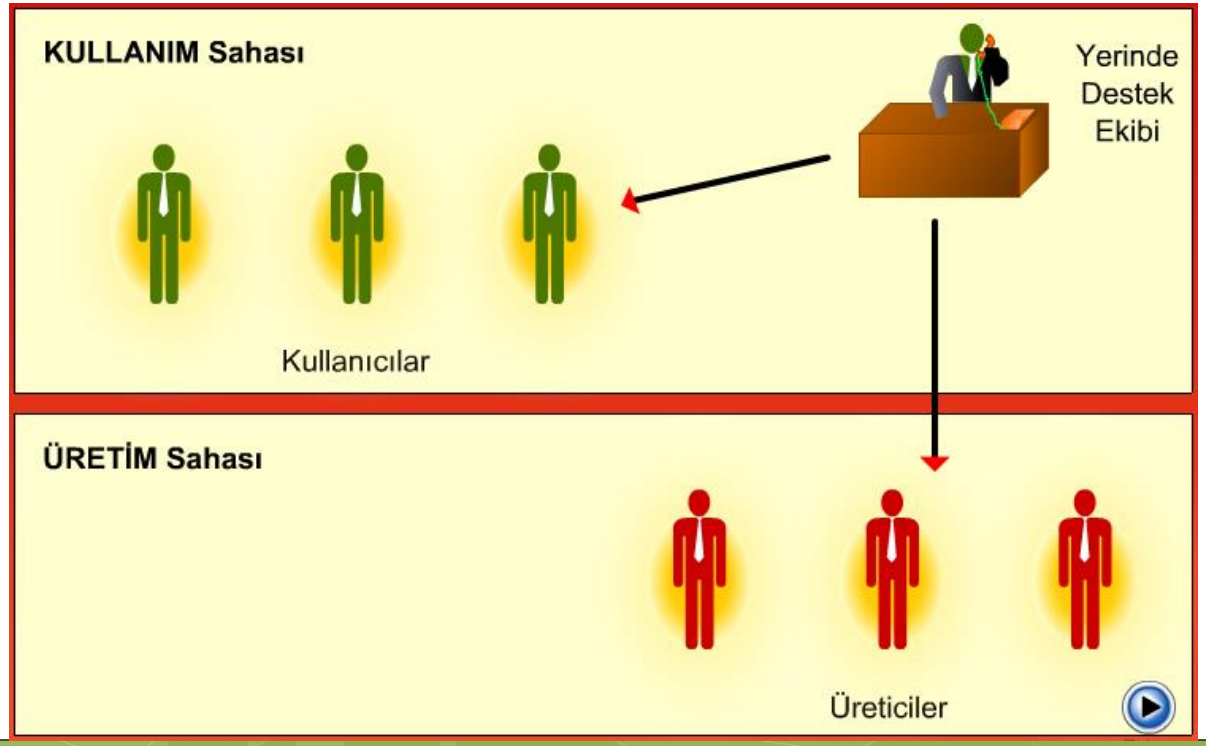
Yerinde Destek Organizasyonu

- Yerinde destek ekibi, kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir. Bu ekibin temel görevleri:

- Kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak,
- Giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasındaki uygulama yazılımı destek ekibine iletmek,

Yerinde Destek Organizasyonu

- Kullanıcıya iş başında uygulama eğitimi vermek,
- Kullanıcı sınama günlüklerini toplamak
- Yapılan tüm işlemleri konfigürasyon veri tabanına kaydetmek biçimindedir.



YAZILIM BAKIMI

1. Tanım
2. Bakım Süreç Modeli
   2.1. Sorun Tanımlama Süreci
   2.2. Çözümleme Süreci
   2.3. Tasarım Süreci
   2.4. Gerçekleştirim Süreci
   2.5. Sistem Sınama Süreci
   2.6. Kabul Sınaması Süreci
   2.7. Kurulum Süreci

başlıkları altında incelenecektir.

1.Tanım

- Bakım, işletime alınan yazılımın sağlıklı olarak çalışması ve ayakta kalabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlanır. Uygulamada çalışan bir yazılımın üç tür bakım gereksinimi bulunmaktadır:
  - Düzeltici Bakım
  - Uyarlayıcı Bakım
  - En İyileyici Bakım

1.Tanım

Düzeltici Bakım

Teorik olarak bir yazılımın tümüyle sınanabilmesi olası olsa bile, pratikte bu sağlanamaz. Bu nedenle çalışan bir yazılımda her an hata ile karşılaşma olasılığı vardır. Bu nedenle zaman zaman çalışan yazılımda ortaya çıkan hataların düzeltilmesi gerekir. Bu tür düzeltme çalışmaları "Düzeltici bakım olarak adlandırılır.

1.Tanım

Uyarlayıcı Bakım

Uygulama yazılımları, işletme ya da kuruluşlarn günlük yaşamlarında yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan araçlardır. Her kuruluş ya da işletme canli bir varlık gibi düşünülebilir. Hiçbir işletme durağan değildir. Süreç içinde değişkenlik gösterir. İşletme ya da kuruluşlarda değişiklik, yapılan işlerin yapılma tarzının değişmesi, yeni iş türlerinin ortaya çıkması biçiminde kendini gösterebilir. İşletme ya da kuruluşlarda yaşanan değişikliklerin, o kuruluşun işlerini bilgisayar yardımı ile yapmalarını sağlayan uygulama yazılımlarına da yansıtılması gerekir. Bu yansıtma işlemi, "Uyarlayıcı Bakım" olarak tanımlanır.

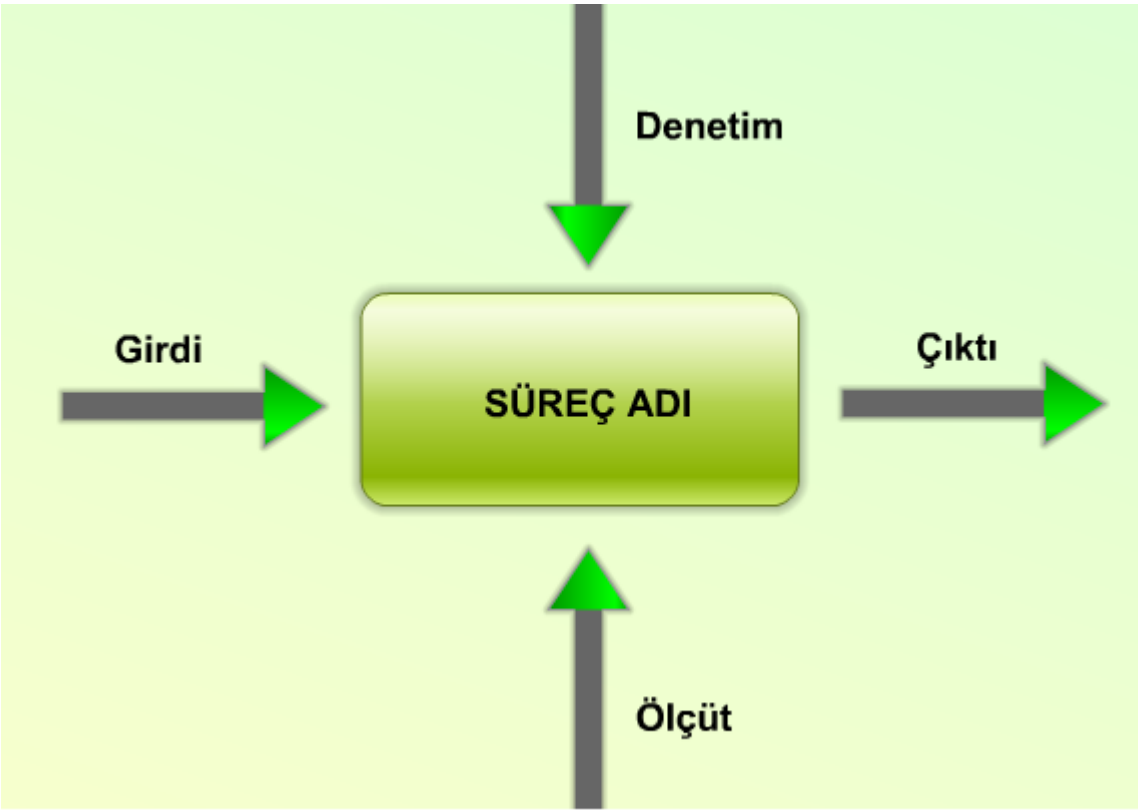
1.Tanım

En İyileyici Bakım

Zaman zaman uygulama yazılımlarının çalışma performanslarının iyileştirilmesi amacıyla yapılan çalışmalar "en iyileyici bakım" olarak tanımlanır.

2. Bakım Süreç Modeli

- IEEE 1219 standardı tarafından önerilen bakım süreç modeli şekilde belirtilen şablonu kullanarak bakım süreçlerini tanımlamaktadır.



2. Bakım Süreç Modeli

Bakım süreç modelinin süreçleri:

- **Sorun Tanımlama/sınıflandırma,**
- **Çözümleme,**
- **Tasarım,**
- **Gerçekleştirim,**
- **Sistem Sınama,**
- **Kabul Sunama**
- **Kurulum** biçimindedir. Görüldüğü gibi, bakım süreci, yazılım yaşam döngüsü çekirdek adımlarının bir anlamda yinelenmesinden oluşmaktadır. Bu yinelenme yalnızca değişiklik isteklerinin var olan koda aktarılması amacıyla yapılmaktadır. Bakım süreçleri aşağıda açıklanmaktadır.

2.1. Sorun Tanımlama Süreci

2.1. Sorun Tanımlama Süreci

Girdi
Sorun tanımlama sürecinin temel girdisi, Bakım isteğidir. Herhangi bir bakım isteğine örnek olarak: Sistemde beklenen ve yeni düzenlemelere ilişkin değişiklikler, • Yeni işlev istekleri, • Yazılımda bulunan yanlışların düzeltilme istekleri, • Performans artımına ilişkin istekler, • Yeni iş yapma türlerine ilişkin istekler ve • Teknolojinin zorlaması sonucu oluşan istekler türündeki istekler verilebilir.

2.1. Sorun Tanımlama Süreci

İşlem/Süreç
Yazılım bakım isteği oluştuğunda yapılması gereken işlemler: **a.** Değişiklik isteğine bir tanım numarası verilmesi, **b.** Değişiklik türünün sınıflandırılması, **c.** Değişiklik isteğinin kabul, red ya da daha ayrıntılı incelenmesi yönünde karar alınması, **d.** Değişiklik ile ilgili zaman/boyut/işgücü kestirimi yapılması, **e.** Değişiklik isteğinin önceliklendirilmesi **f.** Değişiklik isteğinin diğerleri ile birlikte zaman ve iş planına kaydedilmesi biçiminde özetlenebilir. Bu işlemlerin birçoğunda, değişiklik isteğini yapan kişi, kullanıcı temsilcileri, yazılım mühendisleri ve konu uzmanları birlikte çalışıp karar verirler.

2.1. Sorun Tanımlama Süreci

Denetim

Sorun tanımlama aşamasında, değişiklik isteğinin daha önceden yapılıp yapılmadığı denetlenmeli ve tek olduğu belirlenmelidir. Bu amaçla, daha önceki değişiklik istekleri taranır.

Çıktı

Bu sürecin temel çıktısı, doğrulanmış ve geçerlenmiş ve karar verilmiş Bakım İsteğidir. Bir veri tabanında saklanan bu isteğin ayrıntıları:

- Sorun ya da yeni gereksinimin tanımı,
- Sorun ya da gereksinimin değerlendirmesi,
- Başlangıç öncelik,
- Geçerleme verisi (Düzeltici bakım için),
- Başlangıç kaynak gereksinimi,
- Mevcut ve gelecekte kullanıcılar üzerindeki etkileri,
- Yararlı ve aksak yönler

2.1. Sorun Tanımlama Süreci

Ölçüt
Sorun tanımlama sırasında kullanılabilecek ölçütler: • Bakım isteklerinde kabul edilmeyen madde sayısı, • Gelen bakım istekleri sayısı ve • Sorun geçerleme için harcanan kaynak ve zaman biçimindedir.

2.2. Çözümleme Süreci

- Çözümleme süreci, veri tabanında saklanmış ve geçerlenmiş bakım isteğini girdi olarak alır, projeye ilişkin bilgi ve belgeleri kullanarak söz konusu isteğin yerine getirilmesi için gerekli genel planı yapar.

2.2. Çözümleme Süreci

Girdi
Çözümleme sürecinin girdileri: • Geçerlenmiş bakım isteği, • Başlangıç kaynak gereksinimleri ve diğer veriler ve • Mevcut proje yada sistem bilgi ve belgeleri biçimindedir

2.2. Çözümleme Süreci

İşlem/Süreç
Çözümleme süreci temel olarak iki aşamadan oluşur: Olurluk aşaması ve ayrıntılı çözümleme aşaması. Mevcut sisteme ya da projeye ilişkin yapısal belgelerin bulunmadığı durumlarda tersine mühendislik yöntemi kullanılır. Olurluk çalışması, • Değişikliğin etkisi, • Prototiplemeyi içeren seçenek çözümler, • Dönüştürme gereksinimlerinin çözümlemesi, • Güvenlik ve emniyet zorunlulukları • İnsan faktörleri, • Kısa ve uzun erimli maliyetler ve • Değişikliği yapmanın yararları bilgilerini içerir. Ayrıntılı çözümleme çalışmasında, değişiklik isteği için ayrıntılı gereksenim tanımlaması yapılır. Bu çalışmada etkilenen yazılım öğeleri (yazılım tanımları, yazılım gereksinimleri, tasarım, kod, vb) belirlenir. Yazılım öğelerinin değişmesi gereken kısımları belirlenir. En az üç düzeyli sınama stratejisi (birim sınaması, bütünleştirme sınaması ve kabul sınaması) tanımlanır. Bu aşamanın son çalışması ise, kullanıcıya en az etki yapacak şekilde değişiklik gereksinimlerinin nasıl karşılanacağı bilgilerini içeren "Başlangıç gerçekleştirim planı" dır.

2.2. Çözümleme Süreci

Denetim

Çözümleme çalışmasının denetiminde aşağıdaki işlemler yapılır.

a. Gerekli proje yada sistem bilgi/belgelerine erişimin sağlanması (Ortam denetim organizasyonundan)
b. Önerilen değişikliklerin ve çözümleme çalışmasının teknik ve ekonomik olurluğunun gözden geçirilmesi,
c. Güvenlik ve emniyet konularının tanımlanması,
d. Önerilen değişikliğin, mevcut yazılımla bütünleştirilmesinin dikkate alınması,
e. Proje belgelerinin düzgün olarak günlendiğinin denetimi,
f. Çözümleme belgelerinin düzgün olarak hazırlanmasının sağlanması,
g. Sınama stratejilerinin uygun olarak belirlenmesi.

2.2. Çözümleme Süreci

Çıktı

Çözümleme çalışmasının çıktıları:

- Değişiklik isteklerine ilişkin olurluk çalışması,
- Ayrıntılı çözümleme raporu,
- İzlenebilirlik listesini içeren günlenmiş gereksinim tanımları,
- Başlangıç değişiklik listesi,
- Sınama stratejisi,
- Gerçekleştirim planı

2.2. Çözümleme Süreci

Ölçüt
Çözümleme çalışmasında kullanılabilecek ölçütler: • Gereksinimlerdeki değişiklik sayısı, • Belgeleme hata oranı, • Her işlev alanı için gerekecek işgücü ve • Toplam zaman biçimindedir.

2.3. Tasarım Süreci

- Tasarım aşamasında, değişiklikten etkilenebilecek tüm proje bilgi ve belgeleri üzerinde çalışma yapılıp söz konusu bilgi ve belgeler değişiklikle ilgili olarak günlenir.

2.3. Tasarım Süreci

Girdi

Tasarım çalışmasının girdileri:

- Çözümleme çalışması çıktıları
  1. Ayrıntılı çözümleme,
  2. Günlenmiş gereksinim tanımları,
  3. Başlangıç değişiklik listesi,
  4. Sınama stratejisi,
  5. Gerçekleştirim planı
- Sistem ve proje belgeleri ve
- Varolan kaynak kodları açıklamalar ve veritabanları

biçimindedir.

2.3. Tasarım Süreci

İşlem/Süreç
Tasarım için gerekli temel işlemler aşağıda belirtilmektedir. • Etkilenen yazılım modüllerinin tanımlanması, • Yazılım modül belgelerinin değiştirilmesi, • Yeni tasarım için, güvenlik ve emniyet konularını da içeren sınama senaryolarının hazırlanması, • İlişki sınamalarının tanımlanması, • Kullanıcı belgelerinin günleme gereksinimlerinin tanımlanması • Değişiklik listesinin günlenmesi

2.3. Tasarım Süreci

Denetim

Tasarım çalışmasında aşağıdaki denetim ortamı kurulmalıdır.
Tasarımın belirlenen standartlara uygunluğunun denetlenmesi,

- Güvenlik ve emniyet konularını içeren yeni tasarım belgesinin tasarım belgesinin ve bilgilerinin oluşturulmasının sağlanması,
- Sınama bilgilerinin günlenmesinin sağlanması,
- Gereksinimlerden tasarıma izlenebilirliğin sağlanası.

2.3. Tasarım Süreci

Çıktı

Bakım tasarımı çalışmasının çıktıları:

- Gözden geçirilmiş değişiklik listesi,
- Günlenmiş tasarım
- Günlenmiş sınama planları,
- Günlenmiş ayrıntılı çözümleme,
- Günlenmiş gereksinimler,
- Gözden geçirilmiş gerçekleştirim planı
- Risk ve kısıtlar listesi

biçimindedir.

2.3. Tasarım Süreci

Ölçüt

Tasarım çalışması için kullanılabilecek ölçütler aşağıda verilmektedir.

- Yazılım karmaşıklığı,
- Tasarım değişiklikleri
- Her işlev alanı için gerekecek işgücü,
- Toplam zaman,
- Sınama yönerge ve plan değişiklikleri,
- Önceliklendirmedeki hata oranları,
- Varolan kodda, eklenen, çıkarılan ve değiştirilen satır sayısı,
- Uygulama sayısı.

2.4. Gerçekleştirim Süreci

- Gerçekleştirim süreci, temel olarak tasarım çıktılarını ve kaynak kodları girdi olarak almakta ve değişiklik isteğini gerçekleştiren kod parçaları ile günlenmiş yazılım kodlarını üretmektedir.
- Günlenmiş yazılıma ilişkin sınama bilgi ve belgelerinin ve eğitim belgelerinin üretimi de bu süreçte yapılmaktadır.

2.4. Gerçekleştirim Süreci

2.4. Gerçekleştirim Süreci

Girdi
Gerçekleştirim sürecinin girdileri: • Tasarım çalışması sonuçları, • Varolan kaynak kodlar, açıklamalar, belgeler ve • Proje ve sistem belgeleri, biçimindedir.

2.4. Gerçekleştirim Süreci

İşlem/Süreç
Gerçekleştirim sürecinin dört ana işlemi vardır: • Kodlama ve birim sınama • Bütünleştirme" Risk çözümleme • Sınama hazırlığı gözden geçirme • Kodlama işleminde, değişiklik isteğini karşılayan yazılım kodları, varolan yazılıma eklenmektedir. İşlem sonucunda elde edilen yeni, değişmiş modüllere birim sınama uygulanmaktadır. Birim sınama işlemini, bütünleştirme sınama izlemekte, tüm sistem yeniden sınanmaktadır.Uygulamadaki riskleri gidermek amacıyla, gerçekleştirim aşamasında sürekli risk çözümleme yapılmaktadır.

2.4. Gerçekleştirim Süreci

Denetim

Gerçekleştirim sürecinde oluşturulacak denetim yapısı, aşağıdaki özellikleri sağlamalıdır:

- Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması,
- Birim ve bütünleştirme sınamaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması,
- Sınama belgelerinin günlenmesi ve oluşturulmasının sağlanması,
- Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin sağlanması,
- Yeni yazılımın, yazılım ortam yönetimi altında kaydedilmesi ve denetlenmesinin sağlanması,
- Teknik ve eğitim belgelerinin günlenmesinin sağlanması,
- Tasarımdan koda izlenebilirliğin sağlanması

2.4. Gerçekleştirim Süreci

Çıktı

Gerçekleştirim süreci aşağıdaki çıktıları vermelidir:

- Günlenmiş Yazılım,
- Günlenmiş tasarım bilgi/belgeleri,
- Günlenmiş sınama belgeleri,
- Günlenmiş kullanıcı belgeleri,
- Günlenmiş eğitim kılavuzları,
- Riskler ve kullanıcılara etkileri,
- Sınama hazırlığı gözden geçirme rapor.

Ölçüt

Gerçekleştirim çalışmasında kullanılabilecek ölçütler:
" Değişiklik oranı ve
" Hata oranı
biçimindedir.

2.5. Sistem Sınama Süreci

- Değişikliklerin varolan yazılıma yansıtılmasından sonra elde edilen yeni yazılım sürümünün belirlenen standartlara uygun olarak tümüyle bütünleşik sistem üzerinde sınamaların yapılması gerekmektedir.
- Sistem sınamalarının, kullanıcı ve üretici ekiplerin tanıklığında bağımsız bir yapı tarafından gerçekleştirilmeleri önerilmektedir.

2.5. Sistem Sınama Süreci

2.5. Sistem Sınama Süreci

Girdi
Sistem sınama sürecinin girdileri: **a.** Sınama hazırlık raporu **b.** Belgeler • Sistem sınama planları, • Sistem sınamaları, • Sistem sınama yönergeleri, • Kullanıcı kılavuzları, • Tasarım **c.** Günlenmiş sistem biçimindedir.

2.5. Sistem Sınama Süreci

İşlem/Süreç
Sistem sınama, tümüyle bütünleşik bir sistem üzerinde yapılmalıdır. Bu aşamada, işlevsel sistem sınama, arayüz sınama, regresyon sınama ve sınama hazırlık raporunun gözden geçirilmesi işlemleri yapılır. Denetim: Sistem sınamaları, üretici ve kullanıcılardan bağımsız bir grup tarafından gerçekleştirilmelidir. Yazılım kodları ve her türlü bilgi belge, yazılım ortam yönetimi tarafından saklanır.

2.5. Sistem Sınama Süreci

Denetim
Gerçekleştirim sürecinde oluşturulacak denetim yapısı, aşağıdaki özellikleri sağlamalıdır: • Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması, • Birim ve bütünleştirme sınamaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması, • Sınama belgelerinin günlenmesi ve oluşturulmasının sağlanması, • Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin sağlanması, • Yeni yazılımın, yazılım ortam yönetimi altında kaydedilmesi ve denetlenmesinin sağlanması, • Teknik ve eğitim belgelerinin günlenmesinin sağlanması, • Tasarımdan koda izlenebilirliğin sağlanması

2.5. Sistem Sınama Süreci

Çıktı

Gerçekleştirim süreci aşağıdaki çıktıları vermelidir:

- Günlenmiş Yazılım,
- Günlenmiş tasarım bilgi/belgeleri,
- Günlenmiş sınama belgeleri,
- Günlenmiş kullanıcı belgeleri,
- Günlenmiş eğitim kılavuzları,
- Riskler ve kullanıcılara etkileri,
- Sınama hazırlığı gözden geçirme rapor.

Ölçüt

Gerçekleştirim çalışmasında kullanılabilecek ölçütler:

- Değişiklik oranı ve
- Hata oranı

biçimindedir.

2.6. Kabul Sınaması Süreci

- Kabul sınaması süreci, kullanıcılar ya da kullanıcı temsilcileri tarafından gerçekleştirilen bir süreçtir. Kullanıcıların, değişiklikleri içeren yeni yazılımı sınamaları ve kabul etmeleri beklenmektedir.

2.6. Kabul Sınaması Süreci

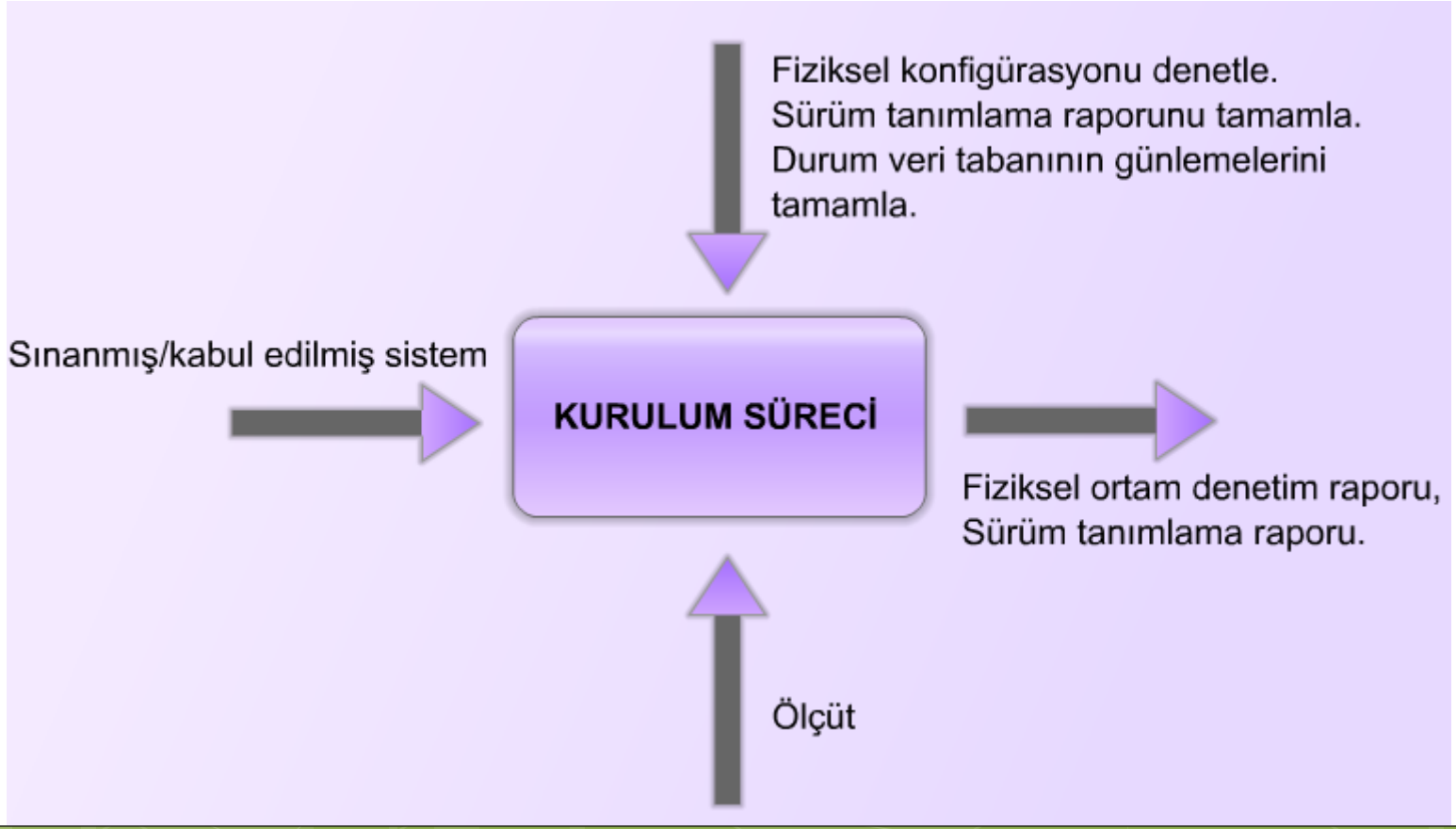
Girdi
Kabul sınama sürecinin girdileri: **a.** Gözden geçirilmiş sınama hazırlık raporu, **b.** Tümüyle bütünleşik sistem, **c.** Kabul sınama planları, **d.** Kabul sınamaları ve **e.** Kabul sınama yönergeleri biçimindedir.
İşlem/Süreç
Kabul sınaması işlemleri: **a.** İşlevsel kabul sınamalarının yapılması, **b.** Birlikte çalışabilirlik sınaması, **c.** Regresyon sınaması biçimindedir.

2.6. Kabul Sınaması Süreci

Denetim
Kabul sınamaları sırasında denetimi aşağıdaki işlemleri içermektedir. a. Kabul sınamalarının uygulanması, b. Sınama sonuçlarının raporlanmasının sağlanması, c. İşlevsel denetim yapılması, d. Yeni sistemin oluşturulması, e. Kabul sınama belgelernin yazılım konfigürasyonuna yerleştirilmesi.
Çıktı
Kabul sınamalarının çıktıları, yeni sistem, işlevsel konfigürasyon denetim raporu ve kabul sınaması raporudur.
Ölçüt
Bu aşamada kullanılabilecek ölçütler: üretilen ve düzeltilen hata oranlarıdır.

2.7. Kurulum Süreci

- Kurulum süreci, geliştirilen ya da değiştirilmiş yeni yazılım sürümünün, uygulama sahasına aktarılma işlemlerini içerir.



2.7. Kurulum Süreci

Girdi
Bu sürecin temel girdisi, tümüyle sınanmış ve kabul edilmiş yeni yazılım sürümüdür.
İşlem/Süreç
Bu sürecin işlemleri: a. Fiziksel ortam denetiminin yapılması, b. Kullanıcıların bilgilendirilmesi, c. Varolan sistemin yedeklerinin alınması, d. Kullanıcı tarafında kurulum ve eğitimlerin yapılması biçimindedir.

2.7. Kurulum Süreci

Denetim
Denetim işlemleri: a. Fiziksel ortam denetiminin yapılması, b. Sistem ile ilgili bilgi ve belgelerin kullanıcıya ulaştırılması, c. Sürüm Tanımlama raporunun tamamlanması ve d. Yazılım konfigürasyon ortamına aktarımın sağlanması alanlarını içermektedir.
Çıktı
Bu sürecin temel çıktıları, fiziksel ortam denetim raporu ve Sürüm tanımlama raporudur.
Ölçüt
Bu süreçte kullanılabilecek ölçüt, belgeleme değişiklikleridir.

Alınan Dersler 1

- Kurulum planlamasında mevsimsel koşulların dikkate alınması önemlidir.
- Örneğin, kış mevsiminde yoğun kar alan doğu bölgelerindeki sahalara kurulum zaman zaman olanaksızdır.
- Bu nedenle, kurulum planlamasında, bu tür bölgeler bahar ya da yaz ayları içerisinde planlanmalıdır.

Alınan Dersler 2

- Kurulum, yalnızca uygulama yazılımı sürümlerinin yüklenmesini içermemekte, zaman zaman teknolojik alt yapı değişebilmekte ve yeni sistem yazılımı sürümlerinin yüklenmesi gerekebilmektedir.
- Bu nedenle, kurulum elemanlarının, gerekli teknik bilgilerle donatılması ve teknolojik değişimleri izlemelerinin sağlanması gerekmektedir.

Alınan Dersler 3

- Uç kullanıcılar zaman zaman kendi bilgisayarlarına dışarıdan getirdikleri ya da internet ortamından sağladıkları yazılımları yüklerler.
- Virüs riski taşıyan bu tür yazılımlara karşı kullanıcıları sürekli uyarmak ve anti-virüs yazılımlarının kullanımını özendirmek gerekmektedir.

Alınan Dersler 4

- Ülkemizde kullanılan yazılım üretim yöntemleri düşünüldüğünde, bilinen anlamda bakım yapılmadığı, yazılımları hazırlayan kişilerin yıllarca kendi yazdıkları yazılıma yamalar yaptıkları, yazılımı usta-çırak ilişkisi içerisinde başkalarına devretmeye çalıştıkları ve çalışan yazılıma ilişkin elde düzgün teknik belgelerin olmadığı bir ortamla karşılaşılmaktadır.

Alınan Dersler 4-5

- Bu durum zaten pahalı olan bilişim iş gücünü verimsiz kullanmamıza neden olmaktadır.
- Kullanıcı ve yerinde destek elemanları arasındaki tüm iletişimin kayıt altına alınmasının sağlanması ve izlenmesi gerekmektedir.

Sorular

1. Kurulum işlemlerini belirtiniz.

2. Yerinde destek ekibinin görev ve sorumluluklarını tanımlayınız.

3. Uygulamakta olduğunuz bir proje için yaptığınız ya da yapmayı düşündüğünüz kurulum işlemlerini açıklayınız.

4. Geçiş dönemini tanımlayınız.

5. Bir uygulama yazılımının ömrü nasıl belirlenir?

6. Yazılım Bakımını tanımlayınız, bakım türlerini açıklayınız.

7. Bakım süreçlerinin her birinde oluşabilecek risklere birer örnek veriniz.

Genel

- Ders Kitabı: Yazılım Mühendisliği

  Erhan Sarıdoğan- papatya Yayıncılık (kitapyurdu.com)

  Diğer Kaynaklar:
  - Ders Notları.
  - Ali Arifoğlu, Yazılım Mühendisliği. SAS bilişim Yayınları
  - İnternet, UML Kaynakları
  - Roger S. Pressman, Software Engineering – Practitioner's Approach

Hafta -1

Dönem sonu raporları IEEE SRS (IEEE Software requirements spesification) Formatında olacak.

Yazılım Yaşam Döngüsü: Planlama-Çözümleme-Tasarım-Gerçekleştirme-Bakım. (P.Ç.T.G.B.)

Yazılım Geliştirme süreci;

- Gereksinim analizi (requirements analysis)
- Spesifikasyon
- Mimari (Architecture)
- Dizayn-Tasarım (Design)
- İmplementasyon (Programlama)
- Test
- Sahaya yerleştirme (Kurulum yada dağıtımda denilebilir)
- Bakım ve Onarım (Maintenance)

User Story (Kullanıcı hikayeleri) demek.

Yazılım gereksinimleri tanım belgesinde (YGTB); geliştirilecek olan yazılımın işlevsel ve işlevsel olmayan gereksinimleri (ihtiyaçları) tanımlanır. Bunlar Use-Case, Kullanıcı arayüzleri, Kısıtlamalar, User Storyler, Sistemin analizleri, Sistemin tasarımı, ER diyagramlarıdır.

PROJENİN AMAÇ KISMI:
Sorun analizi ve hedef analizi bu bölümde yazılır. Projenin kapsamını, tanımlarını, kısaltmalarını ve referansları (kaynakları) yine bu bölümde yazılır.

Proje dökümanları; Times New Roman 12 punto 1,5 satır aralığı ve iki yana yaslı şekilde olmalı.

Bir projede bulunması gereken bölümler ve açıklamaları:

1.bölüm- Tanıtım: Projenin amacı, kapsamı (hangi soruna çözüm bulduğu) ve tanıtımı yapılır.

2.bölüm- Planlama: Yazılım yaşam döngüsünün ilk fazı, planlamadır. Bu kısımda iş-zaman diyagramı olan GANT diyagramı çizilir. Ekip yapısı şematik olarak verilir. Bu bölümde insan kaynaklarının, donanım kaynaklarının ve yazılım kaynaklarının planlanması yapılır.

3.bölüm- Çözümleme: Bu bölümde mevcut sistemin analizi, incelenmesi ve sistemin yetemediği, eksik kalan noktalarını nasıl tamamlayacağımızın belirtilmesi gerekiyor. Use case diyagramları kullanılıyor.

İhtiyaçları belirlerken arayüzler, kısıtlar, use-case ve user storyleri kullanarak belirliyoruz. Arayüzler bu noktada önemli.

4.bölüm- Tasarım: Sistemin dizaynı, mantıksal tasarımı bu bölümde yapılır. Sistem içinde kullandığımız veritabanları, veri akışları, aktivity diyagramları ile sistemin arka planında ne olduğunu tasarlanır.

5.bölüm- Gerçekleştirme: Önerilen sistem gerçekleştirilirken hangi programlama dili kullanıldı, hangi araçların hangi teknolojilerin neden seçildiği bu bölümde açıklanır. Veritabanı sisteminde hangi mimari kullanıldığı bu bölümde açıklanır.

6.bölüm- Test: Yazılım yaşam döngüsünün her fazının doğal bir aktivitesi olarak tanımlanmıştır. Bu bölümde test planı verilir yani doğrulama ve geçerleme işlemlerinin nasıl yapılacağına ilişkin gant diyagramı olarak verilir.

7.bölüm- Bakım: Yazılım kurulum aktiviteleri ve sonrasında nasıl destek verileceği tarif edilmelidir.
8.bölüm- Sonuç: Projede neler yapıldığı, neler kazanıldığı eksik kalan kısımların özet geçildiği bölüm.
9.bölüm- Kaynaklar: Proje sürecinde kullanılan kaynaklar numaralandırılarak yazılır. Hangi sırada olduğu, nerede kullanıldığı, hangi kaynaktan alındığı yazılır.

PROJE YÖNETİMİ

Proje nedir: Belirli bir yerde (belirli bir sahada) belirli bir zaman ve bütçe çerçevesinde, belirli bir mali doğrultuda, bir başlama ve bitiş noktası olan, önceden belirlenmiş amaçların ulaşılmasını sağlayacak olan faaliyetler topluluğudur.

Proje neleri sağlamalıdır:

- Belirli bir yerde uygulanmalı yani uygulama alanı olmalı.
- Belirli bir takvimi zaman aralığı olmalı
- Bütçesi olmalı
- Ve tanımlanmış faaliyetleri olmalıdır.

Her fikir bir proje değildir. Bir fikrin proje olabilmesi için, şu sorulara cevap verebilmeli.

- Nerede uygulayacaksınız.
- Ne kadar sürede bitireceksiniz
- Belirli bir yer ve sürede gerçekleştireceğiniz faaliyetler için ne kadar mali kaynağa ihtiyaç var.
- Bu sürede ve yerde hangi faaliyetleri gerçekleştireceksiniz.

Amaç dökümantasyonu: Proje ile ortadan kaldırmayı, iyileştirmeyi önerdiğimiz sorunlara işaret etmelidir. Bunun için Sorun analizi ve Hedef analizi yapmak önerilir. Amaç, Sorun ve Hedef analizini içermelidir.

Sorun Analizi: Projenin çözmekte rol oynayacağı/katkı sağlayacağı temel sorunun belirlenmesi ve alt sorunların da tanımlanmasıdır. "İlgili projenin temel sorunu ne?" sorusuna ve alt sorulara cevap arar.

Hedef Analizi: Sorun analizinde belirlenen temel sorunları ve alt sorunları olumluya çeviren önermeler ortaya koymaktır. Sorun çözümüne yönelik hedefler konulur.

Projenin genel amacı; Temel sorunun çözümüne yönelik bir hedef, **özel amaçları ise** alt sorunların çözümüne yönelik hedeflerdir. Özel amaçların proje sonunda gerçekleştiğinin ispat edilmesi gerekir.

Özel amaçlar, Faaliyetler ile karıştırılmamalıdır. Örneğin bir eğitimin gerçekleştirilmesi, bir makinanın alınması FAALİYETTİR. Eğitim sonucunda katılımcıların teknik kapasite, bilgi ve becerilerinin arttırılması, alınan makine ile üretimde artış sağlamak ise faaliyete bağlı özel amaçlardır.

Araştırma Süreci:

Bir projede araştırma yapılmak istenirse süreçler sırasıyla;

- Öncelikle literatür taraması yapılmalı. (Arama motorlarından şu ana kadar yapılan projeyle alakalı tarama yapmak)
- Bu alandaki çalışmalar okunup notlar çıkarılmalı
- Literatür taraması sonrası, araştırma sorusu belirlenmeli.
- Veriler toplanmalı
- Analiz yapılmalı
- Analiz sonrası elde edilen bulgular yorumlanmalı
- Son olarak proje raporu hazırlanmalı

Proje seçiminde cevap aranan sorular…

- Hangi alanlara katkı sağlayacak
- Ne düzeyde katkı sağlayacak
- Ne kadar değerlidir
- Ne kadar anlamlıdır
- Hangi soruna/sorunlara nasıl çözümler bulmaktadır
- Özgün ve yenilikçi yaklaşım var mıdır?
- Kim/kimler için önemlidir. HEDEF KİTLESİ KİM?
- Geliştirilebilir yanları var mıdır?

BAZI UML ve DİYAGRAM ÇİZİM ARAÇLARI

Argo UML-Java tabanlı çizim aracı. Tüm UML diyagramları (Sınıf diyagramları, Nesne diyagramları, Aktivity ve Sequence diyagramları) çizilebilir. Çizilen UML diyagramları için kod üretimi mümkündür.
Altova U Model: Veri yönetimi, yazılım ve uygulama geliştirme, mobil ve veri entegrasyonu konularında yardımcı olan bir araç. Altova şirketi tarafından geliştirilmiş.
Visual Paradigm
IBM Rational Software Architect
Microsoft Office Visio. Microsoft tarafından geliştirilmiş olan
Online olarak çizim yapılabilecek program (Bulut ortamında): **Gen My Model**

BAZI PROJE YÖNETİM ARAÇLARI

Redmine: Web tabanlıdır. Ruby on Rails web çatısı ile geliştirilmiş. Bünyesinde GANT (iş takip) çizelgesi barındırır.
Taiga:
Jira: Hata izleme ve Çevik (Agile) proje yönetimine izin veren, çevik metodolojiye uygun.
Apache Subversion: 2000 yılında Collabnet firması tarafından başlanıp desteklenen bir sürüm kontrol sistemidir.

Kullanılacak Yazılımlar

Proje raporlama ve dökümantasyon işlemlerinde ihtiyaç duyulabilecek bazı araçlar şöyledir;

- MS Office Word (proje raporu yazımı)
- MS Office Excel (Tablolama vb.)
- Adobe Photoshop (şekil çizimlerinde vb.)
- IBM RSA(Retional Software Architect)(UML diyagram çizimlerinde)
- MS Visio(Akış şeması çizimlerinde vb.)

Yazılım Geliştirme İstatistikleri

- Tipik yazılım projesinin geliştirilmesi 1-2 yıl sürüyor ve en azından 500.000 kod satır içeriyor
- Tüm projelerin yalnız %70-80'i başarıyla tamamlanıyor
- Tüm geliştirme sürecinde her birey günde ortalama 10 satırdan az kod yazıyor
- Geliştirme süresince her 1000 kaynak kod satırında 50-60 hata bulunuyor (satışa sunulmuş sistemde hata sayısı 4/1000'e düşüyor)

Yazılım Problemleri(Krizleri)

- Tasarlanan zamanın gerisinde kalma
- Bütçeyi aşma
- Düşük Kalite
  - Güvenilir olmayan yazılım
  - Kullanıcı taleplerinin karşılanmasında yetersizlik
  - Sürekliliğin sağlanmasındaki zorluk

Yazılım-Donanım Evrimi

- Erken Yıllar 1950-1960 ilk bilgisayarlar, makine dili, 3-4 şirket
- Yazılımlar; kullanıcı ile birebir iletişimde bulunmayan, işlerin toplu olarak verilip, yalnızca yazıcı çıktılarının alındığı biçimde geliştirilmekte idi.
- Ayrıca, yazılımlar bu günkü gibi ürün tarzında değil, kuruluşa özel olarak geliştirilmekte idi.

Programlama Dillerinin Seviyeleri

1.Kuşak- Makine Dili

10101110 10010001

2. Kuşak- Assembler

8085, Z80, 68000, vs.

3.Kuşak - Üst Seviye Diller

Pascal Coral66 Basic

Bildirimsel

LISP Hope Prolog

Nesneye Yönelik Diller

Smalltalk, C++, Java

4. Kuşak- Veri Yapısal

CICS, SQL

5. Kuşak- Yapay Zeka
Ve Paralel Programlama

CSP, OCCAM

! Tarihten bugüne süreç içerisinde donanım maliyetleri azalsa da, bakım ve idame maliyetleri artmaktadır. Geliştirme maliyetleri de zaman zaman geçmişle aynı maliyette veya artmaktadır.

Yazılım =Mantık(algoritma) +
Veri(test verisi, bilgi) +
Belge (dokümanlar) +
İnsan (kullanıcı, geliştirici) +
Program (kod)

“Bilgisayar sisteminin donanım bileşenleri dışında kalan her şey”

<u>Yazılım</u>

Mantık, veri, belge, insan ve program bileşenlerinin belirli bir üretim amacına yönelik olarak bir araya getirilmesi, ve yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.

Mantık (algoritma)

- Bilgisayarlaştırılmak istenen işin mevcut mantığı yazılıma yansıtılmak durumundadır.

- Bu nedenle mantık (algoritma)bileşeni yazılımın en önemli bileşenlerinden biridir.

Veri

- Her tür yazılım mutlaka bir veri üzerinde çalışmak durumundadır.

- Veri dış ortamdan alınabileceği gibi, yazılım içerisinde de üretilebilir.

- Yazılımın temel amacı veriyi bilgiye dönüştürmektir.

Belge (dokümanlar)

- Yazılım üretimi bir mühendislik disiplini gerektirir.
- Mühendislik çalışmalarında izlenen yol ya da kullanılan yaklaşımlar yazılım üretimi için de geçerlidir.
- Yazılım üretimi sırasında, birçok aşamada yapılan ara üretimlere ait bilgiler (planlama, analiz, tasarım, gerçekleştirim, vb. bilgileri) belli bir düzende belgelenmelidirler.

! İşletmeler doğaları gereği dinamik bir yapıya sahiptir ve zaman içerisinde sürekli olarak yeni istek ve gereksinimler ortaya çıkabilmektedir. Bu yüzden yazılım geliştirirken “yeniden kullanılabilirlik” oldukça önemli bir unsurdur.

Yazılım vs Donanım

- Yazılım geliştirilir donanım üretilir.
- Yazılım eskimez.
- Yazılım en az donanım kadar önemlidir.
- Yazılım kopyalama ve donanım kopyalama farklıdır.

<u>IEEE'nin yazılım mühendisliği tanımı</u>

"Yazılım Mühendisliği:
Sistemli, düzenli, ölçülebilir bir yaklaşımın yazılım geliştirmede, yazılımın işlenilmesinde ve bakımında uygulanmasıdır.
• D*iğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.*

Yazılım Hataları

Mantıksal Tasarım	%20
İşlevsel Tasarım	%15
Kodlama	%30
Belgeleme ve Diğerleri	%35

! Kalite sistemli bir proje, sanılanın aksine kalite sistemsiz bir projeye göre daha az masraflıdır.

Yazılımda Hata Düzeltme Maliyetleri

Yazılım üretimindeki hatalar yayılma özelliği gösterir.

Analiz	1
Tasarım	5
Kodlama	10
Test	25
Kabul Testi	50
İşletim	100

Yazılımda Kalite

Yazılımda kalite sağlama etkinlikleriyle

- Yazılım maliyetleri düşürülür
- Yazılım üretiminin yönetimi kolaylaşır
- Belgeleme ve standart sorunları giderilir.

Ekonomi	Tamlık	Yeniden Kullanılabilirlik	Etkinlik	Bütünlük
Güvenirlik	Modülerlik	Belgeleme	Kullanılabilirlik	Temizlik
Değiştirilebilirlik	Geçerlik	Esneklik	Genellik	Sınanabilirlik
Taşınabilirlik	Bakılabilirlik	Anlaşılabilirlik	Birlikte Çalışabilirlik	

- Yazılım Yaşam Döngüsü
  - Herhangi bir yazılımın, **üretim ve kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
  - Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
  - Müşteri istekleri sınırsız olduğu için, özellikle **BAKIM** aşaması bu döngüyü kendi içerisinde barındırır.
  - Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur. Bu nedenle yazılım yaşam döngüsü **tek yönlü ve doğrusal değildir.** Tabi burada hangi geliştirme modeline göre işlettiğimiz de önemlidir.
  - **Yazılım yaşam döngüsü için oluşturulmuş standart** → IEEE 12207

- Yazılım Yaşam Döngüsü Çekirdek Aşamaları
  - **Planlama → Personel ve Donanım Gereksinimlerinin çıkartıldığı ve fizibilite(yapılabilirlik) çalışmasının yapıldığı aşamadır. PROJE PLANI**
  - **Çözümleme(Analiz) → Sistem gereksinimlerinin çıkartıldığı aşama.2 Aşama vardır. 1- Mevcut sistemin analizi 2-Önerilen sistemin modellenmesi.**
  - **Tasarım → Yazılım sisteminin temel yapısının oluşturulduğu aşamadır.**

    1-Mantıksal tasarım; önerdiğimiz sistemin yapısı anlatılır. Projenin gerçekleştirilmesine başlamadan önce mimari yapısının ortaya konulduğu faz.

    2-Fiziksel tasarım; Yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.

  - **Gerçekleştirim(implementasyon) → Kodlama, Test etme ve Kurulum çalışmaları.**

  - **Bakım → (Tüm döngüyü içerisinde barındırır) → Hata giderme ve yeni eklentiler yapma (teslimden sonra)**

- Yazılım Yaşam Döngüsü Ne Sağlar?
  - Mühendislik faaliyetlerinin yazılıma uyarlanması**(üretim, işletme ve bakım)**
  - Üretimden kullanıma kadar tüm sürecin planlanması ve tüm sürecin fazlara ayrılması → **(Yönetim Kolaylığı)**
  - Her bir aşamada ne yapılacak?
  - Her bir aşamada ki standartlara uyumluluk → **(Kaliteli yazılım)**

- Yazılım Yaşam Döngüsünün Gerçekleştirilmesi Amacıyla
  - **Belirtim (specification) yöntemleri** → Bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler.

  - **Süreç (process) modelleri** → Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada nasıl uygulanacağını tanımlayan modeller kullanılır.

- Belirtim Yöntemleri
  1. **Süreç Akışı İçin Kullanılan Belirtim Yöntemleri**
     - Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler → (Veri akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları)

  2. **Süreç Tanımlama Yöntemleri**
     - Süreçlerin iç işleyişini göstermek için kullanılan yöntemlerdir. → (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).

  3. **Veri Tanımlama Yöntemleri**
     - Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemlerdir. → (Nesne-İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü)

- **Yazılım Süreç Modelleri**
  - **Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir**
  - **Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.**

    - ✓ **Gelişigüzel Model -→Öğrenci Ders Projeleri**
    - ✓ **Barok Modeli →Kullanım kılavuzu gerektiren basit projeler**
    - ✓ **Çağlayan(Şelale) Modeli →Hastane Sistemleri, Kontrol Sistemleri**
    - ✓ **V Modeli →CRM (MÜŞTERİ İLİŞKİLERİ YÖNETİMİ) PROJE**
    - ✓ **Helezonik(Spiral) Model →Askeri savunma yazılım sistemleri**
    - ✓ **Prototipleme →Yüksek maliyetli projeler için prototipleme.**
    - ✓ **Evrimsel Model→Bankacılık uygulamaları**
    - ✓ **Artırımsal Model →Öğrenci bitirme projeleri**
    - ✓ **Araştırma Tabanlı Model → Satranç, en büyük asal sayı**

- **Gelişigüzel Model**
  - **Herhangi bir model ya da yöntem yok.**
  - **Geliştiren kişiye bağımlı (belli bir zaman sonra o bile anlayamaz.)**
  - **Geliştirme güçlüğü yaşanabilir.**
  - **İzlenebilirliği ve bakımı oldukça zor.**
  - **1960'lı yıllarda kullanılan ilkel model**
  - **Genellikle tek kişilik üretim ortamı**
  - **Basit programlama**
  - **Yazılım yaşam döngüsünde ki fazlar da henüz belirgin değil.**

- **Barok Modeli**
  - **Yazılım yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.**
  - **1970'li yıllarda kullanıldı.**
  - **Belgelemeyi ayrı bir süreç olarak ele alır, yazılımın geliştirilmesi ve testinden hemen sonra ele alır.**
  - **Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.**
  - **Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.**
  - **Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir.**

- **Çağlayan (Şelale) Modeli**
  - **Yazılım yaşam döngüsü temel adımları baştan sona en az bir kez izlenerek gerçekleştirilir.**
  - **İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.**
  - **Geleneksel Model olarak da bilinen bu modelin günümüzde kullanımı giderek azalmaktadır.**
  - **Barok Modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.**
  - **Barok modeline göre geri dönüşler iyi tanımlanmıştır.**
    - ✓ **Bu Modelin Avantajları(Çağlayan- Şelale-Geleneksel)**
      - **Müşteriler ve son kullanıcılar tarafından iyi anlaşılabilen adımlardan oluşur.**
      - **İterasyonlar (Tekrarlamalar) bir sonraki ve bir önceki adımlarla gerçekleşir, daha uzak adımların olması nadirdir.**
      - **Değişiklik süreci yönetilebilir birimlere bölünmüştür.**
      - **Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.**
      - **Kalite gereksinimlerinin, bütçe ve zamana kıyasla çok daha önemli olduğu projelerde iyi çalışır.**
      - **Proje yöneticileri için kolaylıklar(işin dağılımını yapma vb.)**

- ✓ **Bu Modelin Sorunları(Çağlayan-Şelale-Geleneksel)**
  - **Gerçek yaşamdaki projeler genelde yineleme gerektirir.**
  - **Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.**
  - **Yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.**
  - **Projeyi yapan ekip mutsuz olur. Kod yazma dışında kalan(işin %80lik) kısmına ağırlık vermemektedirler.**
  - **Üst düzey yönetimlerin projeyi bir gider merkezi olarak görmeleri(Ürünü görme süresinin uzun oluşu nedeniyle)**
  - **Projenin bitmeyeceği düşüncesini yaygınlaştırır.**

❖ **V Süreç Modeli**

- **Sol taraf üretim, sağ taraf sınama işlemleridir.**
- **Her aşamada testi ön plana çıkaran bir model.**

  - ✓ **Kullanıcı Modeli**
    - **Geliştirme sürecinin kullanıcıyla olan ilişkileri tanımlanmakta. Sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.**

  - ✓ **Mimari Model**
    - **Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.**

  - ✓ **Gerçekleştirim Model**
    - **Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar gerçekleştirilir.**

- **Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygundur.**
- **Bu model, kullanıcın projeye olan katkısını arttırmaktadır.**
- **BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur;**
  - ✓ **İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmaktadır.**

  - ✓ **İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modelleri tasarlanıp, gerçeklenmektedir.**

- **V Süreç Modeli Avantajları;**
  - ✓ **Verification ve validation planları erken aşamada vurgulanır.**
  - ✓ **Verification ve validation sadece son üründe değil, tüm teslim edilebilir ürünlerde uygulanır.**
  - ✓ **Proje yönetimi tarafında takibi kolaydır.**
  - ✓ **Kullanımı kolaydır.**
- **V Süreç Modeli Dezavantajları;**
  - ✓ **Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan sağlamaz.**
  - ✓ **Aşamalar arasında tekrarlamaları kullanmaz. Risk çözümü yok.**

- **Helezonik (Spiral) Model**
  - **En önemli özelliklerinden biri RİSK ANALİZİ OLGUSUNU ön plana çıkarmak ve PROTOTİP yaklaşımını öngörmesi.**
  - **Her döngü bir fazı ifade eder. Doğrudan tanımlama, tasarım vs. gibi fazı yoktur.**
  - **Yinelemeli arttırımsal bir yaklaşım vardır.**

I. **Planlama: Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürünle bütünleştirme.**

II. **Risk Analizi: Risk seçeneklerinin araştırılması ve risklerin belirlenmesi.**

III. **Üretim: Ara ürünün üretilmesi.**

IV. **Kullanıcı değerlendirmesi: Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler.**

- ✓ **Helezonik Model Avantajları;**
  - **Kullanıcı Katkısı**
    - **Yazılımı kullanacak personelin sürece erken katılması, ilerde oluşabilecek istenmeyen durumları engeller.**
  - **Yönetici Bakışı**
    - **Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla süreç boyunca karşılaştıkları için, daha kolay izleme ve hak ediş planlaması yapılır.**
  - **Yazılım Geliştirici(Mühendis) Bakışı**
    - **Yazılımın kodlanması ve sınanması daha erken başlar.**

- **Prototipleme**
  - **Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.**
  - **Gereksinimler netleştikçe prototip kullanıcı ihtiyacına göre düzeltilir.**
  - **Müşteri memnun olana kadar düzeltmelere devam et.**

  - ✓ **Prototipleme – Avantajları**
    - **Kullanıcı sistem gereksinimlerini görebilir.**
    - **Karmaşa ve yanlış anlaşılmaları engeller.**
    - **Yeni ve beklenmeyen gereksinimler netleştirilebilir.**
    - **Risk kontrolü sağlanır.**

  - ✓ **Prototipleme – Dezavantajları**
    - **Belgelenmesi olmayan hızlı ve kirli prototipler(quick and dirty)**
    - **Prototip hedefleri net değilse kod hackleme ya da jenga başlar.**
    - **Düzeltme aşaması atlanırsa düşük performansa yol açar.**
    - **Müşteri prototipten de son ürün gibi görünüm ve etki bekler.**

- **Evrimsel Geliştirme Süreç Modeli**
  - **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 sahiptir.**
  - **Pilot uygulamada kullan, test et, güncelle diğer birimlere taşı**
  - **Modelin başarısı ilk evrimin başarısına bağımlıdır.**
  - **İki çeşit evrimsel geliştirme vardır.**
    - **I. Keşifçi Geliştirme (Exploratory Development)**
      - **Hedef: Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmek**
      - **İyi anlaşılan gereksinimlerle başlanmalıdır.**
      - **"Ne istediğimi sana söyleyemem ama onu gördüğümde bilirim." Yaklaşımı.**

    - **II. Atılacak Prototipleme (throw-away prototyping)**
      - **Hedef: Sistem gereksinimlerini anlamak.**
      - **Tam anlaşılmamış gereksinimlerle başlar.**

  - ✓ **Aksaklıkları(Evrimsel Model)**
    - **Değişiklik denetimi**
    - **Konfigürasyon yönetimidir.**
    - **Kalite yönetimi**
    - **Sürüm yönetimi**

- **Artırımsal Geliştirme Süreç Modeli**
  - **Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.**
  - **Öğrencilerin bir dönem boyunca geliştirmeleri gereken programlama ödevlerinin 2 haftada bir gelişiminin izlenmesi(Bitirme tezleri)**
  - **Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.**
  - **Bir taraftan kullanım, diğer taraftan üretim yapılır.**

- **Araştırma Tabanlı Süreç Modeli**
  - **Yap – At prototipi olarak da bilinir.**
  - **Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.**
  - **Yapılan 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.**
  - ✓ **Örnek Olarak**
    - **En hızlı çalışan asal sayı test programı**
    - **En büyük asal sayıyı bulma programı**
    - **Satranç programı**

- **Metodolojiler**
  - **Metodoloji: Bir BT Projesi ya da yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütünüdür.**
  - **Bir metodoloji;**
    - **Bir süreç modelini,**
    - **Belirli sayıda belirtim yöntemini içerir.**

  - **Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır.**

- **Bir Metodojide Bulunması Gereken Temel Bileşenler**
  - **Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (ISO, IEEE vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.**
  - **Standart ve rehberler sürekli olarak güncellenmektedir.**
  - **Bir kuruluşun kendi metodolojisini geliştirmesi oldukça kapsamlı, zaman alıcı ve uzmanlık gerektiren bir faaliyet olup. İstatistikler 50KİŞİ/AY 'LIK bir iş gücü gerektiğini göstermektedir.**
    - **BİR METODOLOJİ ÖRNEĞİ**
      - **Yourdan yapısal sistem tasarımı metodolojisi;**
      - **Kolay uygulanabilir bir model. Günümüzde yaygın.**
      - **Çağlayan modelini temel almaktadır.**
      - **Bir çok CASE aracı tarafından doğrudan desteklenmektedir.**

4. HAFTA

Günümüzde yazılım projelerinin durumu: %4-5 oranında tam başarı

Başarısızlığın ana sebepleri:

- Müşterinin isteklerini doğru analiz edememek
- Proje için uygun ekibi kuramamak
- Yanlış teknoloji ve mimari seçimleri
- Geleneksel yöntemlerin eksiklikleri
- Müşteriyle iletişimden kaçınmak vs.

%26: Vizyon eksikliği
%24: Değişime direnç
%17: Planlama hataları
%10: Yanlış teknolojiler
%9: Ekip değişiklikleri
%8: Yanlış hedefler
%3: Bütçe hataları
%3: Diğer

*Tüm bunlar çevik yazılımla çözülebilir

Çevik Yazılım Yöntemi:

Çok önemsediği noktalar:	/ / /	**Az önemsediği noktalar:**
Bireylerle Etkileşim		Süreç Ve Araçlar
Çalışan Bir Yazılım		Detaylı Belgelendirme
Müşterilerle İşbirliği		Sözleşmedeki Kurallar
Değişikliklere Uyum Sağlama		Belirli Bir Plan

- Hızlı, devamlı ve kullanışlı yazılım üreterek müşteri memnuniyeti sağlamayı amaçlar.
- Geliştiriciler ile iş adamları arasında günlük ve yakın işbirliği bulunmalıdır.
- Çalışan yazılım gelişimin en önemli ölçüsüdür.
- Taleplerdeki geç değişikliklerin de memnuniyetle karşılanır.
- Yüz yüze görüşme iletişimin en güzel yoludur.
- Kendi kendini organize eden takım yapısı gereklidir.
- Basitlik önemlidir.

Temel Prensipler:

- ✓ Müşteriyi memnun etmek
- ✓ Değişen ihtiyaçları karşılamak
- ✓ Sık aralıklarla ürün teslimi yapmak
- ✓ Yüz yüze iletişime önem vermek
- ✓ Sürdürülebilir gelişmeyi desteklemek
- ✓ Teknik mükemmeliyete, iyi dizayna ve sadeliğe odaklanmak
- ✓ Kendi kendine organize olan takımlar kurmak

Çevik model takımları; kendi kendilerine organize olmuş, hedefleri net, işlerine odaklı, teslim edilebilecek düzeyde ürün ortaya koyabilen 3-7 kişilik gruplardan oluşur. Özellik bazlı takım oluşturulur.

Geleneksel model vs. Agile

Geleneksel model: Müşteriler ne istediğini iyi bilir. Geliştiriciler neyi ne şekilde üreteceklerini iyi bilir. Bu yol boyunca hiç bir şey değişmeyecektir.
Agile (Çevik): Müşteriler ne istediğini keşfeder.Geliştiriciler neyi nasıl üreteceğini keşfeder. Bu yol boyunca birçok değişiklik yapılabilir.

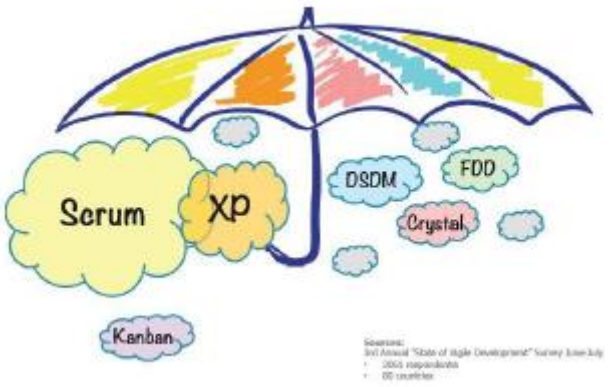
*Geleneksel yöntemde başarı oranı %14 iken Çevik yöntemde %42'dir.

3. Geleneksel Model vs. Agile

Ölçüm	Çevik Modelleme	Çağlayan Modeli
Planlama ölçeği	Kısa dönemlik	Uzun dönemlik
Müşteri ile geliştirici arasındaki mesafe	Kısa	Uzun
Özelleştirme ve uygulama arasındaki zaman	Kısa	Uzun
Sorunları keşfetmek için zaman	Kısa	Uzun
Proje tamamlanma riski	Düşük	Yüksek
Değişikliklere uyum yeteneği	Yüksek	Düşük

3 somut değişiklik yapın>

1) Gerçek takımlar oluşturun
2) Sık sık teslimat yapın (projenin tüm çeyrekleri sonunda)
3) Gerçek kullanıcıları dahil edin.



FDD: Feature – Driven Development

RUP: Rational Unified Process

DSDM: Dynamic System Devolopment Method

FDD (Özellik güdümlü geliştirme): Basit iyi tanımlanmış sistem iyi çalışır. Süreç adımları basit olmalıdır. Sistemi hazırlamak için gereken sistem büyüyebilir olmalıdır. Büyük projeler için de kullanılabilir olmalıdır. İyi süreç arka plana saklanır ve insanlar sonuçlara odaklanabilir. Kısa, iteratif, özellik yaklaşımlı yaşam döngüleri en iyi sonucu verir.

RUP: IBM tarafından oluşturulan bir iteratif yazılım geliştirme süreçlerinden biri. Tümleşik modelleme yapısına uygun. Başarısız bir yazılımdaki sorunların aşılıp başarılı yazılım oluşturmak için gerekli adımları saptayarak oluşturulmuş bir süreçtir.

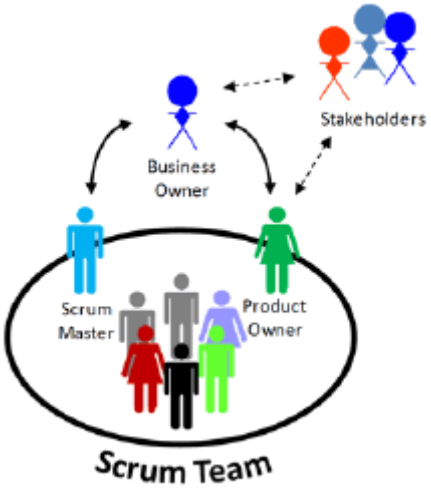
- Müşteriyi ve yazılımcıyı organize edebilmesi,
- Standart tanımlı adımları olması,
- Oluşacak yazılımdaki sık değişiklikleri öngörebilmesi,
- Basit olması,
- Proje yönetim aktivitelerinin çok fazla olmaması. (Sadece geliştirme ön plana çıkar.)

XP (Extreme Programming -- Uç Programlama): yazılım geliştirme süreci boyunca son derece kaliteli olmak koşuluyla çalıştırılabilir kod üretmeye odaklanmış bir yazılım geliştirme metodolojisidir. Sürecin en başından itibaren çalıştırılabilir kodu sürecin merkezinde tutmaktadır.

- Planlama
- Sık ve küçük sürümler
- Basit tasarım:
- Önce test
- Refactoretme
- Haftada 40 saat çalışma
- Müşteriyle yakın iletişim
- Kodlama standartları

~SCRUM MODELİ~

Scrum, çok karmaşık ürün yönetimini basite indirgemek için yapılandırılmış bir yöntemdir. Takım yönetiminde geliştiriciye görünür olması gerek. Şeffaf yönetim söz konusu. Takım yönetiminde ortak bir görüş ortaya koyulur. Kullanıcılarına yönelik istenmeyen sapmaları tespit etmek için sürecin ve programın sürekli denetim altında tutulması gerekiyor.
Scrum takım modeli esneklik, yaratıcılık ve verimliliği optimize etmek için tasarlanmıştır.
Takım kendi kendini örgütler.



Scrum Master: Takım lideri. Uzun vadeli ürün planlamaları. Takımın verimliliğini arttırmakla görevlidir. Gün başında 15dk' lık toplantılar düzenler.

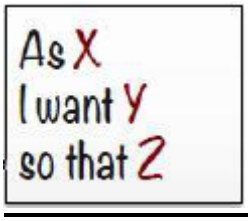
Product Owner: Ürünün geri bildirimlerinden (backlog) sorumlu. Geliştirme ekibi onun kararlarına göre hareket eder.

Backlog:

- Müşteriden ve son kullanıcıdan gelen gereksinimleri içerir.
- "Ne yapacağız" sorusunun yanıtını içerir.
- Herkese açık ve herkes tarafından müdahale edilebilir.
- Risk, iş değeri, zaman gibi kavramlara göre ürün sahibi tarafından sıralandırılır.
- User Story' lerden oluşur.

Sprint: Bir aylık proje döngülerine verilen isimdir. Sprint süresi boyunca her gün toplantılar yapılır. Toplantılarla içerik belirlenir. Sonunda ortada değeri olan bir çıktı olmalıdır.

User Story: Müşteri, son kullanıcı veya ürün sahibi için değerli olan ve anlam ifade eden genellikle fonksiyonel özelliklerin belirtildiği ifadelerdir.

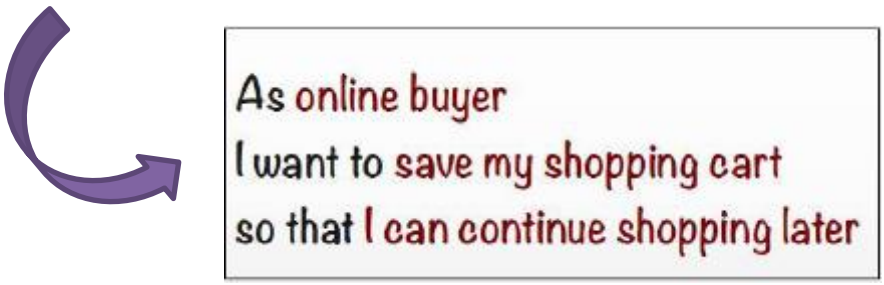


X kişisi olarak

Z olsun diye

Y 'yi istiyorum

Örnek User Story: Online alışveriş yapan biri olarak, alışverişe daha sonra devam edebileyim diye, alışveriş kartımın kaydedilmesini istiyorum.



SCRUM	XP (EXTREMEPROGRAMMİNG)
Sprint (2 hafta – 1 ay)	Sprint (1 ya da 2 hafta)
Sprintler en son halini aldıktan, toplantı yapıldıktan sonra değişmez.	Sprintler değişebilir.
Özellikler geliştiriciler tarafından derecelendirilir.	Özellikler ürün sahibi tarafından derecelendirilir.
Herhangi bir mühendislik pratiği tanımlamaz.	Mühendislik pratikleri tanımlar. Eşli programlama, otomatik test, basit dizayn vs.

5.HAFTA
*En Önemli aşamalardan birisidir(planlama)
*Yazılım geliştirme sürecinin ilk aşaması
* Proje planlama aşamasında yapılan işlemler
1.Proje Kaynaklarının belirlenmesi
2. Proje maliyetlerinin kestirilmesi
3. Proje ekip yapısının oluşturulması
4. Ayrıntılı proje planı yapılması
5. Projenin izlenmesi
*Proje planı tüm proje süresince kullanılabilecek gözden geçirilebilecek ve güncellenecek bir belgedir
1.PROJE KAYNAKLARI
*Zaman kullanımını görev sürelerini edinilme zamanlarını planlar ve 3 grupta incelenir
1.1 İnsan kaynakları : elemanların hangi süreyle ve hangi aşamada olacağını belirler
1.2 Donanım kaynakları : ana bilgisayarlar, sunucular(hep, e-posta, veritabanı) kurumsal firmada bu sunucular farklı makinelerden nedeni a trafiği engellemek için PC LAN WAN
1.3 Yazılım kaynakları : bilgisayar destekli tasarım (CAD)ve bilgisayar destekli mühendislik (CASE) araçları olarak bilinmektedir
*İş sistemleri planlama araçları
*Proje yönetim araçları
*Analiz ve tasarım araçları
*Programlama araçları
*Test araçları
*Simülasyon araçları
*Bakım araçları ve destek araçları
2 PROJE MALİYETLERİNİN KESTİRİLMESİ
*Bir bilgi sistemi yada yazılım için gerekebilecek işgücü ve zaman maliyetlerinin üretiminden önce belirlenebilmesi için yapılan işlemlerdir
KULLANILAN UNSURLAR
*Geçmiş projelere ilişkin bilgiler
*Proje ekibinin deneyimleri
*İzlenen geliştirme modeli
MALİYET YÖNETİMİ SAYESİNDE;
*Gecikmeler önlenir
*Bilgi sistemi geliştirme süreci kolaylaştırılır
*Daha etkin kaynak kullanımı sağlanır
*İş Zaman planı etkin olarak gerçekleştirilir
*Ürün sağlıklı olarak fiyatlandırılır
*Ürün zamanında ve hedeflenen bütçe sınırları içerisinde bitirilir
GÖZLEMLENEBİLECEK DEĞERLER;
*Projenin toplam süresi
*Projenin toplam maliyeti
*Projede çalışan eleman sayısı
*Toplam satır sayısı
*Bir satırın parasal maliyeti
*Bir kişinin ayda gerçekleştirdiği satır sayısı
*Toplam işlev noktası sayısı
*Bir işlem noktasının parasal maliyeti
*Bir ayda gerçekleştirilen işler noktası sayısı

*Bir kişinin bir ayda gerçekleştirdiği işler noktası sayısı
*Kişinin maliyeti
İŞLEV NOKTALARI YÖNTEMİ
İşlev noktaları, geliştirmenin erken aşamalarında saptanan bir değerdir ve 3 temel adımda izlenir
1. problemin bilgi ortamının incelenmesi
2. Problemin teknik karmaşıklığın incelenmesi
3. İşlev noktası hesaplama
1 .PROBLEMİN BİLGİ ORTAMININ İNCELENMESİ
Bu aşamada problemle ilgili temel girdiler, çıktılar, kütükler, sorgular ve arayüzler incelenir problemin bilgi ortamı, 5 ana bileşenden oluşur
*Kullanıcı girdileri: yazılıma girdi olarak verilen her farklı uygulama bileseni bir kullanıcı girdisi olarak sayılır kullanıcı girdilerini sayısı hesaplanırken alan bazında değil daha genel olarak mantıksal kayıt bazında uygulama yapılmalıdır
Örneğin bir personel sisteminin geliştirilmesinden söz edildiğinde personel sicil bilgilerini personel izin bilgilerini gibi bilgiler aynı kullanıcı girdiler olarak sayılabilir
*Kullanıcı Çıktıları : kullanıcıyı ilgilendiren her türlü mantıksal çıktı kullanıcısı olarak sayılmalıdır bu kapsamda kullanıcı çıktılarını örnek olarak raporlar ekran çıktıları hata iletileri ve benzeri verilebilir girdiler gibi çıktılar da genel olarak sayılmalıdır bir raporun içerisindeki bir alan kullanıcı çıktısı olarak sayılmamalıdır öte yandan aynı bilgileri içeren ancak bilgi düzeni ya da sıralaması farklı olan raporlarda ayrı kullanıcı çıktısı olarak ele alınmamalıdır
Örneğin soyadına göre sıralanmış personel listesi ve sicil nosuna göre sıralanmış personel listesi raporları farklı kullanıcı çıktılar olarak sayılmamalıdır
*Kullanıcı sorguları: kullanıcı sorgusu çevrimiçi olarak bilgisayarlara verilen bir kedi sonuncu yine çevrimiçi olarak bir kullanıcı çıktı saldırması biçiminde tanımlanmaktadır her farklı sorgu ayrı olarak sayılmalıdır
Örneğin personel sicil bilgilerinin sorgulanması personel maaş bilgilerinin sorgulanması vb.
*Kütükler: her mantıksal bilgi yıgini ya da kütük ayrı olarak sayılmalıdır. Personel sicil kütüğü, personel maaş kütüğü personel hareket kütüğü vb.
*Arayüzler: geliştirilmesi öngörülen bilgi sisteminin, gerek kurum içinde gerekse kurum dışında bir başka bilgi sistemi ile bilgisayar ortamında bir iletişimi söz konusu ise bu durum bir dışsal arayüz olarak sayılmalıdır kağıt ya da rapor türündeki arayüzler bir dışsal arayüz sayılmazlar
Örneğin, personel Bilgi sistemi, yine aynı kurumda olduğu varsa elden muhasebe bilgi sistemine, personel maaşlarına ilişkin bilgiyi bilgisayar ortamında aktarılması bir dışsal arayüz olarak sayılabilir
*Problem bilgi ortamı için elde edilen değerler , karmaşıklığı düzeylerine göre seçilecek karmaşıklık değeri ile çarpılır ve çarpımlarının sonucu elde edilen toplam işlem nokta sayısı, ayarlanmış işlevnokta sayısı (AİN) olarak adlandırılır
2. PROBLEM TEKNİK KARMAŞIKLIĞININ İNCELENMESİ
Bu adımda, probleme ilişkin dışsal etkenlerin değerlendirilmesi yapılır problem teknik karmaşıklık faktörü problemin karmaşıklığına ilişkin sorulara verilecek yanıtların toplamı ile elde edilir uygulama ortamının ve uygulamanın özellikleri dikkate alınarak her soruya 0 ile 5 arasında bir yanıt verilir(TKF)
3.İŞLEV NOKTA SAYISI HESAPLAMA
İN =AİN*(0.65*0.01*TKF)
İŞLEV NOKTA SAYISI ÇEŞİTLİ AMAÇLARLA KULLANILABİLİR
ÜRETKENLİK = İN/KİSİ-AY
KALİTE=hatalar/İN
ETKİN MALİYET MODELİ COCOMO
Tüm COCOMO modelleri, temel girdi olarak satır sayısı kestirimini alır ve çıktı olarak işgücü ve zaman çıktılarını verir.

İş gücü K =a*S^b
Zaman=c*K^d
a,b,c,d: her biri model için farklı olan katsayılar
S=1000 türünden satır sayısı
Öte yandan COCOMO modelleri farklı türdeki yazılım projelerinin farklı biçimde uygulanır projeler 3 biçimde sınıflandırılmaktadır
1. AYRIK PROJELER
Deneyimli ve küçük proje ekipleri tarafından geliştirilecek, boyutları küçük bilgi sistemi projeleri dir bu tür projelerde, Bir donanımın yazılım tarafından sürülmesi ya da yönetilmesi söz konusu değildir genellikle, yerel ağ üzerinde çalışan, girdi ve çıktıları tümüyle veri bilgi tabanlı olan projelerdir örneğin, bir kurumun yerel ağ üzerinde çalışan insan kaynakları yönetim sistemleri vb.
K=2.4*S^1.05 işgücü
T=2.5*K^0.38
2. YARI GÖMÜLÜ PROJELER
hem bilgi boyutu hem de donanım sürme boyutu olan projelerdir
3. GÖMÜLÜ PROJELER
Donanım sürmeyi hedefleyen ve donanım kısıtları yüksek projelerdir (pilotsuz uçağı yönlendiren yazılım projeleri vb.
PROJE PLANI
1. PROJE KAPSAMI: geliştirilecek sistemin genel tanımını ve sınırlarını içerir yalnızca neyin yapılacağı değil nelerin yapılmayacağı da proje kapsamında yer almalıdır
2. PROJE ZAMAN-İŞ PLANI : projede yapılacak kişilerin iş adımları, ara ürünlerin neler olacağı zamanlaması ve ne zaman üretileceği, kimler tarafından üretileceği türündeki bilgiler proje zaman iş planında yer alır bu plan proje geliştirme döngüsü sürecinde neler yapılacağını zaman boyutuyla içerir proje zaman iş planını hazırlamak amacıyla geliştirilmiş birçok araç ve yöntem bulunmaktadır bu araçlardan en yaygın olarak kullanilanına örnek olarak MS project verilebilir
3. PROJE EKİP YAPISI: proje oluştururken ne tür insan kaynaklarına ihtiyaç var bu kısımda bunlar belirlenir
4. ÖNERİLEN SİSTEMİN TEKNİK TANIMLARI: proje planında ,projede kullanılacak donanım, altyapısı, işletim sistemi, veri tabanı yönetim sistemi vb. Yazılımlara ilişkin özellikler önerilen sistemin teknik tanımları bölümünde yer alır varsa, özel amaçlı donanımlar ve bunların çizimleri, kablolama, işletim zamanı kısıtları gibi gerekler ayrıca belirtilir
5. PROJE STANDARTLARI , YÖNTEME METODOLOJİLERİ: projenin geliştirilmesi amacıyla kullanılacak standartlar, yöntemlerle mitolojiler olacak yanında yer almalıdır söz konusu standart, yöntem ve metodolojileri örnek olarak
*çözümleme aşamasında kullanılacak yöntemler
*Tasarım aşamasında kullanılacak yöntemler
*Belgeleme standartları ve tasarım standartları vb. Verilebilir bu konuda en önemli kaynaklardan biri İEEE standart ve rehberleredir
6. KALİTE SAĞLAMA PLANI: projede yapılan işlerin doğruluğunun denetlenmesi, proje standartları ve izleme yöntemlerinin oluşturulması, proje kalitesinin sağlanması amacıyla kalite sağlama planı oluşturulur bu plan ayrıca projede oluşturulacak kalite yönetim biriminin görev ve iş tanımlarını yapar ana ilke olarak her ara ürün ancak kalite denetiminden geçtikten sonra iş sahibi ya da müşteriye verilir
7. KONFİGÜRASYON YÖNETİM PLANI: sistemin geliştirilmesi sırasında herhangi bir aşamada oluşabilecek değişikliklerin kaydedilmesi, izlenmesi için kullanılacak yöntem yönergeler

konfigurasyon yönetim planında yer alır bu planda temelde müşteri ya da bir iş sahibine değişikleri bildirme ve izleme konusunda yardımcı olması amacıyla hazırlanır
8. KAYNAK YÖNETİM PLANI: projenin , geliştirme süresi boyunca fiziksel kaynakların insan kaynaklarının ve finansal kaynakların yönetimi sürekliliğini sağlama biçimi ile ilgili konular kaynak Yönetim planı içerisinde yer alır
9. EĞİTİM PLANI: büyüklüğü ne olursa olsun her projede eğitim boyutu içerir bir proje için sağlanacak en düşük düzeydeki eğitim geliştirilen uygulama yazılımları ile ilgili olarak uç kullanıcıların eğitimidir bu eğitimin yanısıra bir proje için gerekebilecek eğitimlere örnek olarak kullanıcılar için temel bilgisayar kavramları Windows Office yazılımları teknik eğitim eğitimleri verilebilir söz konusu eğitimlerin ne zaman kime hangi ortamda verileceği gibi bilgiler eğitim planında yer alır
10. SINAMA PLANI: her proje planı geliştirme süresinde üretilecek ürünün gelecek teknik açıdan gerekse kullanıcı açısından sınanacağına alınacağına ilişkin bilgileri icermek durumundadır aksi durumda projenin sonlandırılması olası olmaz söz konusu bilgiler proje sınama planı içerisinde yer alır
11. BAKIM PLANI: proje uygulama aşamasını geçtikten sonra gerekebilecek değişiklikleri ne zaman ve nasıl yapılacağını içerir

SİSTEM ÇÖZÜMLEME

SİSTEM ÇÖZÜMLEMEYE GİRİŞ

-Üretim sürecinin başlangıcıdır. Mevcut sitemin nasıl çalıştığı araştırılır.

-Mevcut sistemin incelenmesi aşamasında temel hedef gereksinimlerin saptanmasıdır. Bu işlemden sonra sistem için modelleme yapılır. Bu modele mantıksal model adı verilir.

-Çözümleme aşamasında iki durum söz konusudur;

1)Mevcut sistemin incelenmesi

2)Önerilen sistemin modellenmesi

-Çözümleme çalışmasında mutlaka bir model takip edilmelidir. Aksi takdirde süreç dağınık bir biçimde sürer ve denetlenemez.

-Bu bölümde yapısal sistem geliştirme yaklaşımında kullanılan yöntemlerden en yaygın olanlarına örnekler verilmiştir.

-Yöntemler veri modelleme ve süreç modelleme olarak iki başlık altında incelenecektir.

GEREKSİNİM NEDİR?

-Bir sistem geliştirirken kullanıcının sistemin işlevleri ile ilgili beklentileri sistemin amaçlarını oluşturur.

-Bu amaçların yerine getirilmesine de gereksinim adı verilir.

-Bir kuruluş için personel bordro sistemi geliştirdiğimizi varsayalım. Maaş formlarının aylık olarak hazırlanması, kuruluşun değişik birimlerinden bu bilgilere erişim istekleri gereksinimlere örnektir.

GEREKSİNİM TÜRLERİ

1)İŞLEVSEL GEREKSİNİMLER

-Sistem ile çevresi arasındaki iletişimi belirleyen gereksinimler olarak tanımlanır. Sistemin herhangi bir durum karşısında durumunu belirleyen fonksiyonel gereksinimler bu başlık altında incelenmektedir.

-Örneğin: Maaş bordro sisteminde; maaş çekinin ne zaman hazırlanacağı, çek hazırlamak için ne tür girdiler verilmesi gerektiği vb.

2)İŞLEVSEL OLMAYAN GEREKSİNİMLER

-Kullanıcının sorununun bağımsız olarak çözülmesi gereken sorunlar işlevsel olmayan gereksinim olarak tanımlanır.

-Daha çok sisteme dair kısıtları ifade eden işlevsel olmayan gereksinimler başlığı altında işlenen ihtiyaçlardır.

-Örneğin; kullanılacak bilgisayarın türü, kullanılacak yazılım geliştirme ortamı, veri tabanı yönetim sistemi vb.

GEREKSİNİMLER

1)FİZİKSEL ÇEVRE

-Fiziksel çevre gereksinimlerinin belirlenmesi için önemlidir.

-Örneğin; işlevlerin geliştirileceği aygıtlar nerededir?

-Tek bir yerde mi olacak birden çok fiziksel birbirinden ayrılmış yerler söz konusu mu?

-Sıcaklık, nem oranı veya manyetik etkileşim gibi çevresel kısıtlamalar var mı?

2)ARAYÜZLER

-Girdiler bir mi yoksa birden çok sistemden mi geliyor?

-Çıktılar bir mi yoksa birden çok sisteme mi gidiyor?

-Verinin nasıl biçimlendirileceğine ilişkin önerilen bir yol var mı?

-Verinin kullanılacağı, önerilen bir ortam var mı?

-Arayüzlerin kullanımı kolay m?

3)KULLANICI VE SİSTEM ETMESİ

-Sistemi kim kullanacak?

-Çeşitli kullanıcılar olacak mı?

-Her bir kullanıcı tipinin yetenek düzeyi nedir?

-Bir kullanıcının sistemi anlaması ne denli kolay olacaktır?

-Bir kullanıcının sistemi kötü amaçla kullanması ne kadar zordur?

4)İŞLEVSELLİK

-Sistem ne yapacak?

-Sistem yapacağı işi ne zaman gerçekleştirecek?

-Sistem nasıl ve ne zaman değiştirilebilir veya güçlendirilebilir?

-Çalışma hızı, yanıt süresi ya da çıktı üzerinde kısıtlayıcı işlemler var mı?

5)BELGELEME

-Ne kadar belgelemenin gerektiği?

-Belgeleme hangi kullanıcı kitlesini hedeflemektedir?

6)VERİ

-Hem giriş hem çıkış için verinin biçimi ne olmalıdır?

-Bu veri kesinlik ölçüsü ne olmalıdır?

-Bu veri ne sıklıkta alınacak veya gönderilecektir?

-Hesaplamalar hangi duyarlılık derecesine kadar yapılacaktır?

-Belli bir zaman süresince veri kaynağında saklanacak mıdır?

7)KAYNAKLAR

-Sistemi kurmak için, kullanılacak ve bakımını yapmak için ne kadar malzeme, personel ve diğer kaynaklara ihtiyaç vardır?

-Geliştirici hangi yeteneklere sahip olmalıdır?

-Sistem ne kadar fiziksel yer kaplayacak?

-Güç, ısıtma ve soğutma için gereksinimler nelerdir?

-Geliştirim için tavsiye edilen bir zaman çizelgesi var mıdır?

-Geliştirme veya yazılım ve donanım için harcanacak para üzerinde bir sınırlama var mıdır?

8)GÜVENLİK

-Sisteme veya bilgiye erişim denetlenmeli midir?

-Bir kullanıcının verisi diğerlerininkinden nasıl ayrıştırılacak?

-Kullanıcı programları diğer program ve işletim sistemlerinden nasıl ayrı tutulacak?

-Yedek kopyaları başka bir yerde mi saklanmaktadır?

-Yangın ve hırsızlığa karşı ne tür önlemler alındı?

9)KALİTE GÜVENCESİ

-Güvenlik için gereksinimler nelerdir?

-Sistemin özellikleri insanlara nasıl aktarılmalı?

-Sistem çökmeleri arasında öngörülecek zaman nedir?

-Bir çöküm sonrasında istemi tekrar başlatmak için uygun görülebilecek bir süre var mıdır?

-Sistem tasarımda yapılacak değişiklikleri nasıl özümseyecek?

-Bakımdan anlaşılan sadece hata gidermek midir yoksa sistemi geliştirmeyi de içerir mi?

-Kaynak kullanımı ve yanıt süresine ilişkin verimlilik ölçütlerinelerdir?

-Sistemi bir yerden bir yere veya bir bilgisayardan diğerine taşımak ne kadar kolaylıkta olmalıdır?

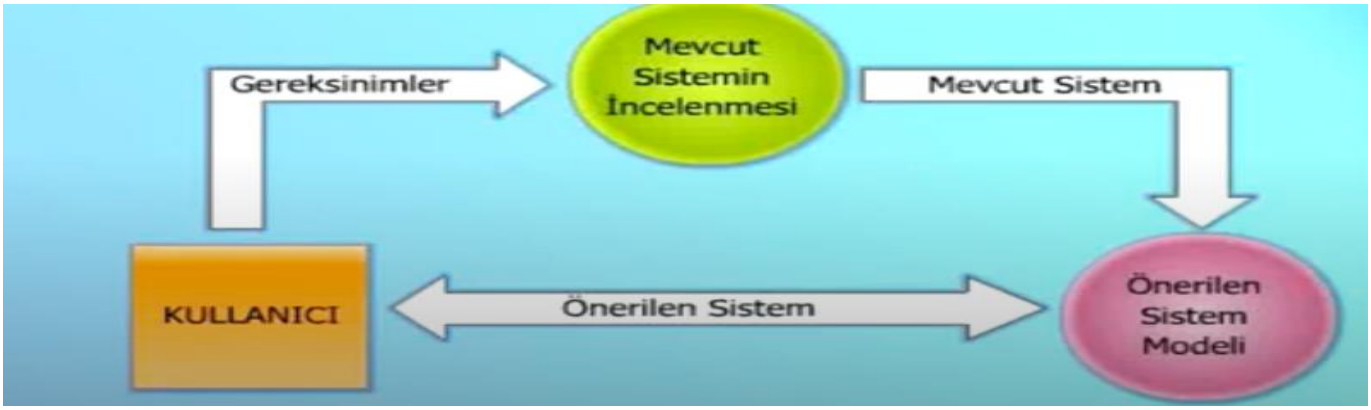
GEREKSİNİM ÖZELLİKLERİ

- Gereksinimler sadece bir sisteme ve sisteme yapılan veri dönüşümünü tanımlaması, aynı zamanda sistemin performansı üzerindeki kısıtları da tanımlar.
- Bu yüzden gereksinimler üç aşamada hizmet eder.
- Birincisi; geliştiricilerin, müşterilerin sistemin nasıl çalışmasını istediklerini anlamalarını sağlar.
- Gereksinimler ayrıca tasarımcılara, sonuç sistemin ne işlevsellik özellikte olacağını söyler.
- Üçüncüsü; gereksinimler, sınama ekibine, kullanıcıyı sunulan sistemin istenen sistem olduğuna ikna etmek için neler göstermeleri gerektiğini söyler.

Hem geliştiriciler hem de kullanıcılar gereksinimleri kullandığı için, gereksinimlerin geçerliliğinin doğrulanması gerekir. Doğrulanma süreci aşağıda belirtilen yedi kriterleri içerir;

1. Gereksinimler doğru oluşturulmuş mudur?
   -Hem geliştirici hem de kullanıcı tarafından belirsizliği giderilmiş hatasız olarak onaylanmış olduğunu güvenceye almak için gözden geçiriniz.
2. Gereksinimler tutarlı mı?
   -Yani, hiç çelişen, çift anlamlı, anlaşılmayan gereksinimler var mı?
3. Gereksinimler tam mı?
   -Eğer tüm olası durumlar bir gereksinimle açıklanıyorsa, gereksinimler tamdırlar. Bu nedenle örneğin bir çalışan ücretsiz izne çıktığında, birisi terfi ettiğinde veya bir kişi avans istediğinde ne olacağı örneğin bir personel sistemi tarafından açıklanmalıdır. Tanımlama, kullanıcı tarafından istenen tüm özellikleri içeriyorsa bir sistem için dışsal tamdır denir. Bir tanımlama; eğer gereksinimler içerisinde açıklanmamış referanslar yoksa içsel tam olarak tanımlanır.
4. Gereksinimler gerçekçi mi?
   -Kullanıcının istediği gerçekten yapılabilir mi? Geliştirme zamanının uzun olduğu durumlarda teknolojik gelişime uyabilecek biçimde tanımlı mı?

5. Her gereksinim kullanıcı tarafından istenen bir şeyi mi tanımlamaktadır?
   -Bazı durumlarda bir gereksinim bizi gereksiz yere kısıtlayabilir. Örneğin kullanıcı ünü iyi olduğu için XYZ bilgisayarının kullanılmasını isteyebilir. Öte yandan XYZ istenen sistemin gerçekleştirilmesinde en uygun bilgisayar olmayabilir. Bu tür gereksinimler doğrudan sistemin hedeflerine adreslemez. Yalnızca işe yarayacak gereksinimler üzerinde yoğunlaşmalıdır.
6. Gereksinimler doğrulanabilir mi?
   -Gereksinimlerin karşılandığını göstermek için sınama senaryoları oluşturulabilir mi? Ya da doğrudan sınama yapılabilir mi? Kullanıcı bir hız tanımlaması yapmadan ve yanıt süresinin koşullarını tanımlamadan hızlı yanıt süresi isteyebilir. Örneğin hızlı yanıt süresi daha doğrulanabilir olarak şu şekilde tanımlanabilir; 'sistemde kelime işlem programı kullanan en çok sayıda kullanıcı ile sistem bir kullanıcının ekranına bir belgeyi bir sonraki sayfasıyla birlikte en çok 5 saniyede getirmelidir.
7. Gereksinimler izlenebilir mi?
   -Her sistem işlevi kendisini zorunlu kılan bir dizi gereksinim olarak izlenebilir mi? Sistemim özel bir yönüyle ilgili bir dizi gereksinim bulmak kolay mı? Örneğin tüm iletişim gereksinimlerini gözden geçirmek için bütün gereksinimler okunmak zorunda mı?



MEVCUT SİSTEMİN İNCELENMESİ

-Bu çalışmada temel amaç yazılım geliştirilecek olan sistemin anlaşılması ve tanımasıdır. bu amaçla görüşme yapma gerekirse anket yapma yöntemleri kullanılır.

- Yapılacak görüşmelerde mevcut sistemde elle yürütülen işlemler girdi, işlev, çıktı ve diğer tüm işlevlerle olan ilişkiler bazında sorgulanır. İlgili yönerge korun ve yöntemler kullanıcıdan edinir.

- Sorgulama ya da inceleme işleminin belirtilmesi amacıyla çeşitli yöntemler kullanılır. sorgulama sonucu elde edilen duygularla ilgili olarak kullanıcıdan geri bildirim alınır.

- Mevcut sistemde elle yapılan işlerde kullanılan form defter ve yazma örnekleri sistemin boyutunu oluşturur. Bu tür bilgilerin birer kopyaları bir sonraki aşamada veri modellemesi için edinilir.

ÖNERİLEN SİSTEMİN MODELLENMESİ

-MANTIKSAL TASARIM-

-Mevcut sistemin modellenmesinden sonra bilgisayarla ortamda işlerin yapılabilmesi amacıyla önerilecek sistemi modeli oluşturulur.

- Bu modelleme daha çok bilgi sistemini geliştirecek teknik personele (sistem tasarımcıları, programlamacıları) yöneliktir. Bu model aynı zamanda mantıksal model olarakta tanımlanır.

- Mantıksal model önerilen sistemin veri yapısını ve süreç yapısını hem genel hem de ayrıntılı olarak tanımlar. Mantıksal model kolaylıkla fiziksel modele (program parçaları veritabanı tabloları vb.) dönüştürülebilir bir yapıdadır.

- Bazı yazarlar ‘mantıksal model’ üretimini tasarım aşamasının bir parçası olarak ele alırlar ve konuyu tasarım taslağı altında incelerler.

7.HAFTA : YAZILIM ÇÖZÜMLEME 2

GEREKSİNİM VERİSİ TOPLAMA:

Temel yöntemler;

1) Sorma

2) Psikolojik Türetme Yöntemi

3) İstatiksel Teknikler

1)Sorma Yöntemi:

* En eski ve en önemlilerinden biri.

* Gereksinim verilerinin toplanması için (İhtiyaçların doğru belirlenmesi için)

* Karşılıklı görüşme / anket

a)Karşılıklı Görüşme:

* Gereksinimlere ilişkin amaç , düşünce, yöntemler, duygu ve düşünceler araştırılır.

* En etkin yollardan biri.

Piramit tarzı: Özel sorulardan genel sorulara doğru.

Koni tarzı: Genel sorulardan özel sorulara doğru.

Elmas tarzı: Özel ile başlayıp genel sorularla sürer ve yine özel sorularla biter.

b)Anket:

* Kullanıcı sayısı fazla ise

* Sistem üzerindeki aktörlerin fazla olduğu durumlarda

* Kullanıcı eğilimlerini ve davranış biçimlerini saptamak amacıyla

* Yazılı test (soru kısmı ve anket kısmı)

2) Psikolojik Türetme Teknikleri :

* Özellikle belirsizlik fazla ise

* Zayıf yapılı ortamlarda

* İnsan psikolojisine dayalı tekniklerle bilgi edinebilmek için

* Anket ve görüşmeye dayalı fakat onlardan farkı “Üçleme Tekniği” ni kullanmasıdır. (Psikolojide bir teknik)

* Algılama haritaları ve neden-etki çizelgeleri kullanmaktadır.

3) İstatistiksel Teknikler :

* Veri yoğunluklu ve veri hacminin yüksek olduğu ortamlarda verinin özelliklerini belirleme amacıyla

* Örnekleme yöntemi ve PIRA Modeli önemlidir.

a) Örnekleme Yöntemi :

*Amaç: veri toplama hızını arttırmak ve verilerdeki çelişkileri önlemektir.

* Basit gelişigüzel örneklem, karmaşık gelişigüzel örneklem, amaçlı örneklem

vb…

* Örneklem boyutu belirli bir güvenilirlik düzeyinde olmak koşulu ile yine

istatistiksel teknikler kullanılarak belirlenir.

b) PIRA Modeli :

* Personal, Interactive, Report and Analysis

* Bilgi gereksinimlerinin tanımlarını belirli normlara bağlı olarak açıklamayı

hedefler. Kişilerin bilgiye dayalı tercihlerini belirlemek amacıyla kullanılır.

VERİ MODELLEME YÖNTEMLERİ :

Sistemin mantıksal modelinde veri yapısını açıklamak amacıyla kullanılmaktadır. Veri yapısını çeşitli düzeylerde tanımlamayı (en soyuttan en ayrıntı düzeye) amaçlar.

En yaygın olanları :

1) Nesne ilişki şemaları

2) Veri sözlüğü

1) Nesne İlişlki Şemaları :

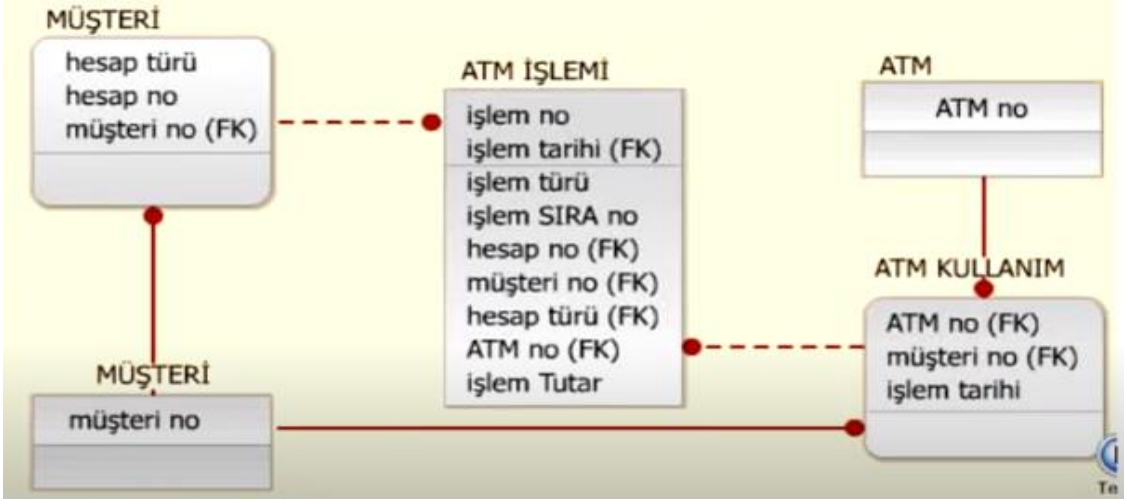
* Bir veri nesnesi 3 temel özelliği ile bilinir:

1) Adı

2) Özellikleri

3) Diğer veri nesnesi varlıklarına referansı (Bu amaçla her bir veri nesnesini tek olarak belirleyen bir belirteç (anahtar) kullanılır. Söz konusu belirteç veri nesnesi varlığının ad özellikleri arasında yer alır.)

* Geliştirilecek sistemin kullancağı ana veri nesneleri ve aralarındaki ilişkiyi belirtir. Örnek :



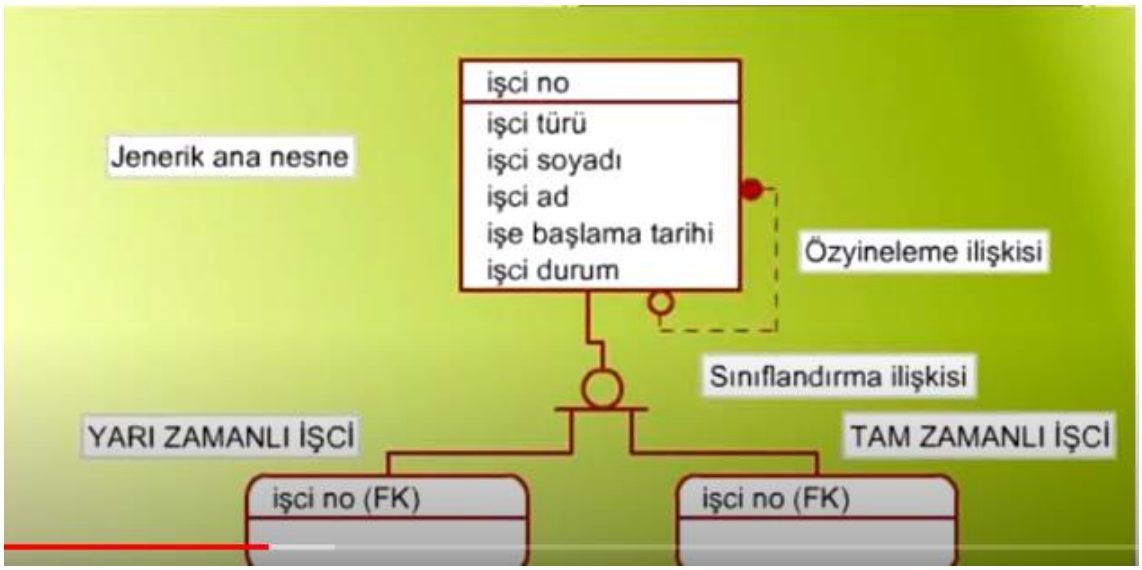
Burada örneğin ATM ve altında özellikleri verilmiştir ya da ATM işlemi ve ilişki içinde bulunduğu ATM kullanımı gibi nesnelerden dolayı foreign key olarak kendi içerisinde bu özelliği almıştır.

Örnek 2 açıklama : Javadan biliyoruz ki nesneler soyut olarak tanımladığımız sınıfların birer gerçek örneğidir. Somutlaştırma adına kullanıyoruz. Burada 2 nesne tanımlanmış biri insan nesnesi diğeri araba nesnesi. İnsan nesnesinin özellikleri olarak çeşitli nitelikler verilmiş (ad, adres, yaş, ehliyet no..). Aynı şekilde araba için de sıralanmış. Burada insan nesnesi le araba nesnesi arasında bir sahiplik ilişkisi vardır.

Örnek 3 açıklama : Burada artık somut olarak bir veritabanı tablosu mevcut. Dolayısıyla biz nesne ilişkisi diyagramlarında kendi veritabanı tablolarımızı oluşturuyoruz.

Marka	Model	Şasi no	Tür	Renk	Sahibi
BMW	740L	XZ68...	Sedan	Siyah	AA
VOLVO	S80	A32XT..	Sport	Sarı	BT
Ford	Taurus	X24R56	Hatchback	Bej	NB

Anahtar Alan



Yine varlık ilişki diyagramlarındakine benzer olarak burada nesneler arasındaki ilişkileri birden bire, birden çoka ve çoktan çoka şeklinde tanımlayabiliyoruz.

1-1 ilişki : Bir insan ancak 1 araba sahibi olabilir.

1-N ilişki : Bir insan birden çok araba sahibi olabilir.

M-N ilişki : Birden insan birden çok araba sahibi olabilir.

VERİ SÖZLÜĞÜ:

Nesne ilişki şemalarında belirtilen nesne özelliklerinin ayrıntılı tanımları veri sözlüğünde yer alır.

* Veri adı,

* Veri eş adı (Aynı veri için kullanılan diğer ad)

* Nerede / nasıl kullanıldığı

* İçerik tanımı

SÜREÇ-İŞLEM MODELLEME YÖNTEMLERİ:

Geliştirilecek sistemin süreç ya da işlemlerini ve bu süreçler arasındaki ilişkileri tanımlamak amacıyla kullanılan yöntemlerdir.

* Veri akış diyagramları (VAD)

* Süreç tanımlama dili (STD)

* Karar tabloları

* Karar ağaçları

* Nesne şemaları

1) Veri Akış Diyagramları (VAD):

* Sistemin mantıksal modeli yukarıdan aşağıya bir yaklaşımla oluşturulur

* Önce en genel biçimiyle (Yalnızca dışsal ilişkiler) sonra iç yapısındaki süreç ve ilişkiler modellenir.

VAD KÜMESİ :

3 tür veri akış diyagramı bulunuyor:

a) Kapsam Diyagramı :

* Dışsal ilişkilerini göstermek amacıyla kullanılır.

* Yalnızca bir çember ve gerekli sayıda veri kaynağı ile aralarındaki veri akışlarının belirtiminden oluşur.

b) Genel Bakış Diyagramı:

* Kapsam diyagramının ayrıştırılmış biçimidir.

* Bilgi sisteminin ana işlevlerini, bu işlevlere ilişkin veri kaynaklarını , veri depolarını ve işlemler arasındaki ilişkileri içeren VAD.

* En çok birkaç tane çizilir.

c) Ayrıntı Diyagramı :

* Genel bakış diyagramında belirtilen her ana işlev, belirli bir ayrıntı düzeyine kadar (en fazla 7+/-2) detay VAD oluşturularak ayrıştırılır.

* Genellikle detay VADlarda veri kaynakları gösterilmez.

VAD Neyi Gösterir?

* Sistemin durağan yapısını

* Süreçlerini ve aralarındaki veri akış ilişkilerini

* Kurum birimlerini ya da dış birimleri bilgi kaynakları olarak gösterir.

* Gerekli olan ana veri depolarının neler olduğunu ve hangi süreçler tarafından kullanıldığını

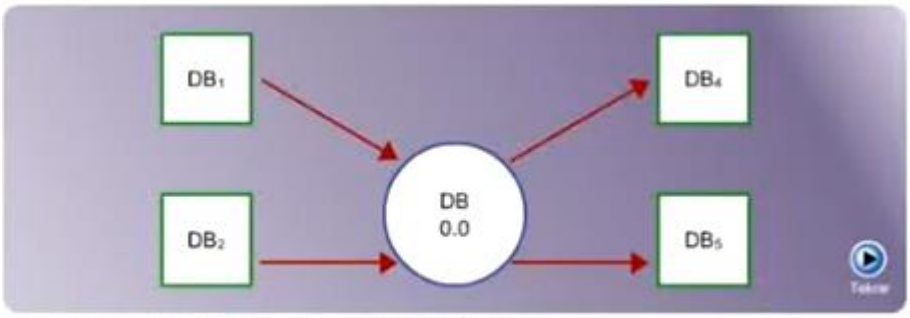
* Süreçleri yukarıdan aşağı doğru ayrıştırarak gösterir. (En genelden en ayrıntı düzeye kadar.)

VAD Neyi Göstermez?

* Zamana ilişkin durumunu ve bu durumla ilgili bilgileri göstermez.

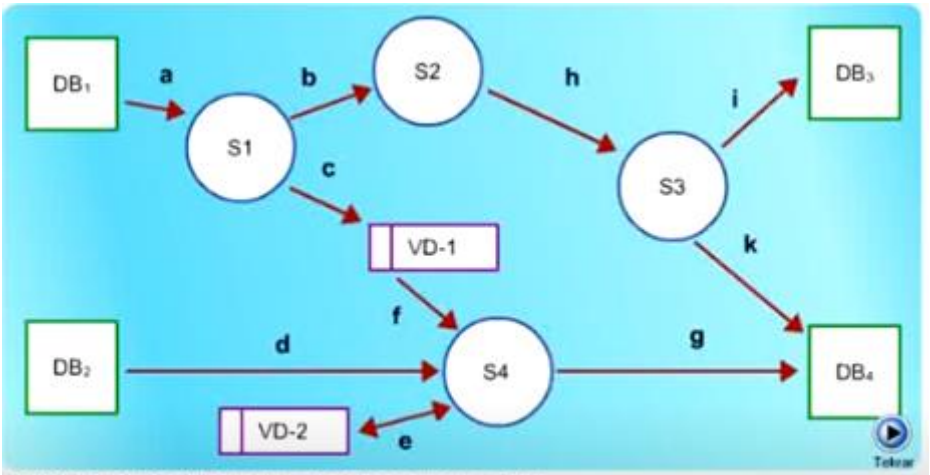
* Bilgi sistemi süreçlerinin kendi aralarındaki “karar” ilişkisini göstermez.

* Gerek bilgi sistemi süreçleri gerek veri akışları gerekse bilgi kaynakları ve bilgi depoları için ayrıntı içermez.



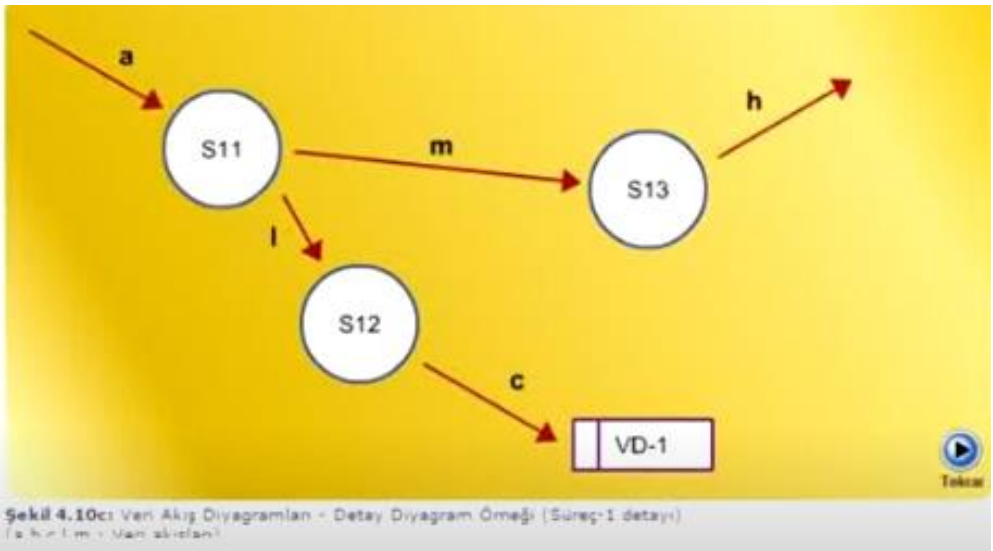
Şekil 4.10a: Veri Akış Diyagramları - Kapsam Diyagramı Örneği
(DB: Dış birim, BS: Bilgi Sistemi)

Örneğin burada bir kapsam diyagramı mevcut. DB dediğimiz bir süreç var ve bu süreçlere bağlı veri kaynaklarımız mevcut. Burada DB herhangi bir dış birimi gösteren bir ifade olarak özellik almış. BS dediğimiz yapılar ise bilgi sistemi olarak özetleniyor. DB1 ve DB2 den bir veri akışı var ve bu veri akışıyla birim dediğimiz yapımız DB4 ve DB5 i verilerle besliyor.



Şekil 4.10b: Veri Akış Diyagramları - Genel Bakış Diyagramı Örneği
(S: Süreç, VD: Veri Deposu, a,b,c,d,e,f,g,h,i,k : Veri akışları)

Genel bakış diyagramı kapsam diyagramlarından biraz daha ayrıntılanmış bir yapı. Burada S dediğimiz yapı süreçtir. VD, veri deposu ve DB ler ise dış birim. Burada yapı biraz daha dağıtılarak süreçler ortaya konmuş.



Süreç1 in detayını veren detay diyagram örneği

2) Süreç Tanımlama Dili :

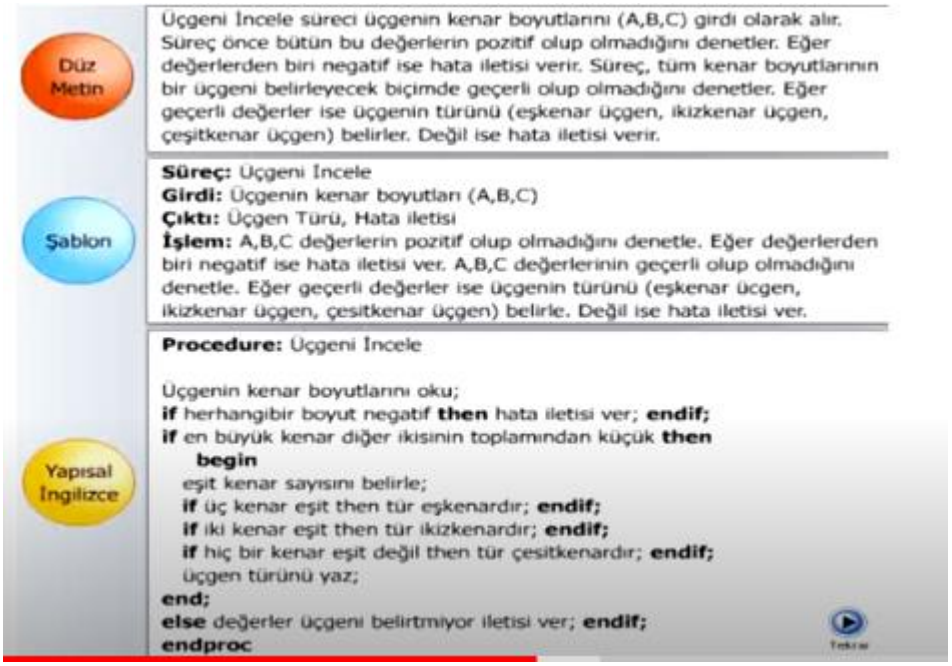
Veri akış diyagramlarında ayrıştırılmış olan bilgi sistemi süreçlerinin iç yapılarını belirtmek amacıyla araç, yöntem ya da gösterim biçimleri :

* Düz metin,

* Şablon,

* Yapısal İngilizce (pseudo koda benzer)

Örnek:



KULLANICI ARAYÜZ PROTOTİPLEME (KAP) :

* Gereksinim tanımlama ya da sistem çözümleme çalışmasının önemli bir bileşeni, oluşturulacak bilgi sistemine ilişkin girdi ve çıktı gereksinimlerinin tanımlanmasıdır.

* Gereksinim çalışmasının hemen sonunda kullanıcıya gösterilecek bir prototip yazılım hazırlanmasını içerir. Söz konusu prototipin gerçekten içsel olarak çalışmayan ancak ekranlar, menüler ve bunların aralarındaki geçişlerin çalıştığı bir yazılımdır.

KAP İçin Ekranlar Nasıl Hazırlanır?

* Hazırlanması için gerekecek iş gücü ve zaman, sistem çözümleme için ayrılan zamanın %5ini aşmamalıdır. (Kod olmaksızın sadece panel, menü vs bileşenlerle hazırlanıyor.)

* Bir özellik birden fazla ekranda yer alsa bile yalnızca bir kez gösterilmelidir.

* Hiçbir kişisel ve içsel işlem içermemelidir.

Örneğin; bir ekranda kişilerle ilgili sorgulama yapıldığını düşünelim. Adı ve soyadı bilinen bir kişinin, bu bilgi verildiğinde, diğer ayrıntı bilgilerinin ekranda görüntüleneceğini varsayalım. Kullanıcı, ad soyad bilgisini girecek ve ilgili sorgulama tuşunu etkinleştirdikten sonra ekranda hep aynı bilgler görünecektir. Yani KAP, Ad soyad bilgisine göre kişi tablosundan arama yapmayacak, program içerisinde hep aynı sonuçları ekrana verecetir. Çünkü kişi tablosunun yaratılması ya da kullanılması söz konusu değildir. Burada amaç, kullanıcıya sorgulama biçimini ve sorgulama sonucunda elde edilecek bilgileri açıklamaktır. Sorgulanan bilginin içeriği bu aşamada önemli değildir. Kullanıcıdan sorgulama sonucu gelen bilgi türlerine ilişkin, sorgulama işleminin işlem akışına ilişkin bilgi alınması hedeflenir.

KAP İçin Raporlar Nasıl Hazırlanır?

* Yazıcı çıktısı biçiminde alınması istenen raporlar bir metin düzenleyici (örneğin MC Word) aracılığıyla hazırlanır ve belirli bir biçimde numaralandırılır.

* Örneğin; bilgi sisteminin insan kaynakları yönetim bilgi sistemi (IKYS) olduğunu varsayalım. Raporları MS Word ile hazırlayıp bunlara “IKYS-AT-R1” , “IKYS-AT-R2” gibi numaralar verelim. Kullanıcıya prototip gösterdiğimizde kullanıcı, rapor alma düğmesini etkinleştirdiğinde “Şu anda yazıcıdan IKYS-AT-R1 nolu rapor dökülmekte.” şeklinde bir ileti alacaktır. Bu sayede kullanıcı bize rapor alanları, raporun tetkiklendiği ekranla ilgili geri bildirimlerinive değişiklik istediğini iletir. Bu değişiklikler yapıldıktan sonra söz konusu raporla ilgili gereksinimler kesinleşmiş olur.

Çözümleme Çalışması Nasıl Değerlendirilir?

Sistem çözümleme çalışması sonuçlandırıldıktan sonra elde edilen ara ürünün (mantıksal model) istenenleri karşılayıp karşılayamadığının belirlenmesi amacıyla değerlendirilir.

Temel olarak ;

*Kullanıcıların benimsemesi ve onayı

* Tutarlılık, tamlık, uygunluk

* Tasarım çalışması için yaterliliği

a) Tamlık ve Tutarlılık :

Tamlık : Tüm ögeler ve bunlar arasındaki ilişkilerin tanımlanmasını gerektirir.

Tutarlılık : Modelin kendi içerisinde hatasız çelişkisiz olmasıdır.

Tamlık ve tutarlılık denetimi, bir programlama dilinde yazılmış bir programın söz dizim kurallarının denetimine benzer.

Örneğin; modelleme aracı olarak VAD kullanıldığında

* Bütün süreçlerin girdi ve çıktıları var mı?

* Süreçler, veri akışları, dış birim ve veri depoları uygun şekilde adlandırılıp numaralandırılmış mı?

* VAD deki tüm isimler veri sözlüğünde yer almış mı?

* Planlama aşamasında öngörülen belirtimlerin hepsi ele alınmış mı?

b) Olurluluk:

*Sistem çözümleme aşamasındaki çalışmanın planlama aşamasında yapılan çalışmaya uygunluğunun belirlenmesi için yapılan çalışmaları içerir.

* Maliyet kestirim çalışması yinelenir, sapmalar saptanır, kaynaklar yeniden planlanır. Sapmalar fazla olursa çalışmanın yeniden gözden geçirilmesi gerekir.

* Bu aşamada maliyet kestirimi için daha fazla bilgi mevcuttur. VAD yöntemi, maliyet kestirim yöntemlerinin kolayca uygulanabilmesini olanaklı kılar.

VAD'den Elde Edilenler:

Girdi ve çıktı sayısı: Doğrudan genel bakış diyagramındaki veri akışlarının incelenmesiyle elde edilir.

Kütük Sayısı: Genel bakış diyagramlarındaki veri depoları sayılarak elde edilir.

Sorgu Sayısı: Genel bakış diyagramı ve detay diyagramlarındaki süreçlerin incelenmesiyle elde edilir.

Arayüz Sayısı: Kapsam diyagramındaki dış birimlerle olan ilişkiler incelenerek elde edilir.

ALINAN DERSLER ÖZET :

* Kullanıcılar, genellikle görüşme tutanaklarını imzalamakta çekingenlik gösterirler.'Siz hazırlayın, bize gönderin bizde imzalar size göndeririz ' biçiminde yöntemler önerirler.

* Görüşme tutanaklarınızı anında iki kopya olarak tutun ve hemen görüşme bitiminde imzalatma yöntemini benimseyin ve uygulayın. Aksi durumda, ileride ortaya çıkabilecek sorunlarda kaybeden taraf siz olursunuz.

* Tüm görüşme ve toplantı kayıtlarınızı olabildiğince bilgisayarlı ortamda saklayın. Aradan, uzun bir süre geçtikten sonra, eski kayıtlara erişimde büyük kolaylıklar sağladığını göreceksiniz.

Bu amaçla Visual Source Safe türü basit bir araç bile kullanabilirsiniz.

* VAD yöntemi, kullanıcı tarafından algılanma açısından oldukça etkili bir yöntemdir. Bu durum, değişik projelerde görülmüştür.

* Kullanıcıya, görüşme ile ilgili olarak düzeylendirilmemiş bir VAD hazırlayıp gönderdiğinizde, kullanıcının bu VAD'nı kendi başına yeniden çizebildiği gözlemlenmiştir.

* CASE aracı kullanımının modelleme çalışmasını oldukça hızalandırdığı gözlemlenmiştir. Niteliği ne olursa olsun mutlaka bir CASE aracı kullanın.

CASE aracı kullanımı ayrıca, daha sonraki aşamalarda oluşabilecek ve mantıksal modele yansıtılması gereken günlemelerin kolayca yapılmasını sağlar.

TASARIM

Bu aşamada yapısal sistem geliştirme metodolojisine göre yapılan bir çözümlemenin tasarıma dönüştürülmesi, tasarım kavramları, veri tasarım ve süreç tasarım yöntemleri açıklanmaktadır.

Tasarım sistem analizi çalışması sonucu üretilen mantıksal modelin fiziksel modele dönüştürülmesi çabasıdır.

Fiziksel model gerektirecek yazılımın;

Hangi parçalardan oluşacağını,bu parçalar arasındaki ilişkilerin nasıl olacağını parçaların iç yapısının ayrıntılarını,gerekecek veri yapısının fiziksel değişimini(dosya,veri tabanı, Hash tablosu, vektör..vs)

tasarımını içerir.

Tasarım Kavramları

Soyutlama (Abstraction):Detayları gizleyerek yukardan bakabilme imkanı sağlar (nesne tabanlı programlama paradigmalarından da biridir.)

İyileştirme(enhancement):Soyutlama düzeyinde irdeleme bittikten sonra,daha alt seviyelere inerek tanımlamalarda ayrıntı, bazen de düzeltme yapılarak tasarımın daha iyi kesinlik kazanması sağlanır.

Modülerlik(modularity)[en önemlisi]:Sistemi istenilen kalite faktörleri ışığında parçalara ayrıştırma sonucu elde edilir

Bütün karmaşıklığın tek bir modülde toplanması yerine,anlaşılabilir ve dolayısıyla projenin zihinsel kontrol altında tutulması için sistem birçok modüle ayrılır.[Hem yönetim olarak kolaylık sağlar hem yazılım karmaşıklığını ortadan kaldırır.]

Modüller, isimleri olan tanımlanmış işlevleri bulunan ve hedef sistemi gerçekleştirmek üzere tümleştirilen birimlerdir.

Sistem modüllerinde dallanan, ağaç veri modülü gibi düşünülebilir. Sistem dallanarak farklı alt modüllere ayrılır.

İşlevsel Bağımsızlık

Modüllere parametreyle değer gönderilir ve sonuç değer alınır.Bu modülü çağıran program parçası sadece bu sonucu kullanabilir.Çağrılan modülün işlevsel olarak yaptıklarıyla ilgili değildir.

Veri Tasarımı

Yapı tasarımı,arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.

Bilgi saklama ve soyutlama bu işlem için önemli kavramlardır.

Yaparken dikkat edilecekler:

Değişik veri yapıları değerlendirilmelidir.

Bütün veri yapıları ve bunlar üzerine yapılacak işlemler tanımlanmalıdır.

Alt düzeyde tasarım kararları tasarım süreci içerisinde geciktirilmelidir.

Bazı çok kullanılan veri yapıları için bir kütüphane oluşturulmalıdır.

Kullanılan programlama dili soyut veri tiplerini desteklemelidir.

Yapısal Tasarım

Ana hedefi modüler bir yapı geliştirip modüller arasındaki kontrol ilişkisini temsil etmektir.Ayrıca yapısal tasarım bazen de veri akışını gösteren biçime dönüştürülebilir.

Veri akışları üç parçada incelenebilir.

Girdi akışı, Çıktı akışı, İşlem akışı

Ayrıntı Tasarım-Süreç Tasarım

Süreç tasarımı veri, yapı ve arayüz tasarımından sonra yapılır.

İdeal şartlarda bütün algoritmik detayın belirtilmesi amaçlanır.

Ayrıca süreç belirtiminin tek anlamı olması gerekir, değişik şahıslar tarafından farklı yorumlanmamalıdır.

Doğal diller kullanılabilir.

Süreç tanımlama dili(PDL)

Yapısal program yapıları

Yapısal programıın temel amacı program karmaşıklığını en aza indirmek ve anlaşılabilirliğini artırmaktır.

Bu amaçla şu yapılar kullanılır:

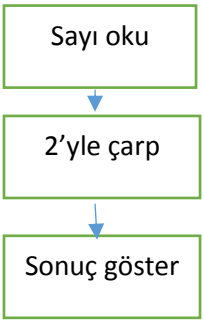
Ardışıl işlem yapısı

Koşullu işlem yapısı

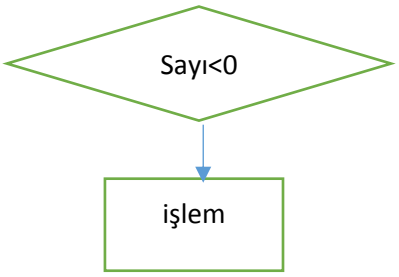
Döngü yapısı

(GOTO (dallanma komutu) kullanımı uygun değildir.)

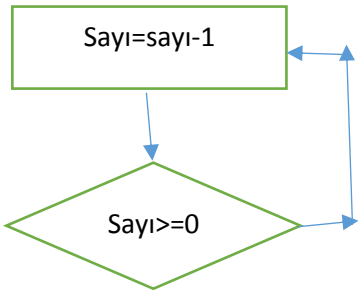
Ardışıl işlem



Şartlı işlem



Tekrarlı işlem



Karar tabloları

Karmaşık koşul değerlendirmeleri yapmak gerekir. Bunların düzenli bir gösterimi karar tablolarında yapılabilir.

Karar Tablosu Örneği

Kurallar	1	2	3	4
Hesap Geçerli	X	X	X	X
Şifre Geçerli	X		X	X
Yeterli Nakit Var	X			
Yeterli Bakiye Var	X		X	
Bakiye Bildirme İşlemi				X
Para Çekme İşlemi	X			
Para Yatırma İşlemi		X		
Para Gönderme İşlemi			X	

Tasarımda Olması Gereken Ortak Alt Sistemler:

Yetkilendirme alt sistemi

Güvenlik alt sistemi

Yedekleme alt sistemi

Veri transferi alt sistemi

Arşiv alt sistemi

Dönüştürme alt sistemi

Yetkilendirme Alt Sistemi

Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabileceği ve kullanamayacakları özellikleri ifade eder.

İşlem bazında yetkilendirme

Ekran bazında yetkilendirme

Ekran alanları bazında yetkilendirme

Güvenlik alt sistemi

Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması işlemleri

LOG Filer (sistem günlüğü)

Yedekleme alt sistemi

Her bilgi sisteminin olağandışı durumlara karşı yedekleme sistemi olmalıdır.

Veri iletişim alt sistemi

Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleri

Çevrimiçi veri iletimi(real-time)

Çevrimdışı veri iletimi(disketler,teypler)

Arşiv alt sistemi

Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemleridir.

Aktif veri sistemi

Dönüştürme alt sistemi

Geliştirilen bilgi sisteminin uygulamaya alınmadan önce veri dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması)işlemlerine ihtiyaç vardır.

Kullanıcı Arayüz Tasarımı

Kullanıcıyla ilişkisi olmayan arayüzler:

Modüller arası arayüz

Sistemle dış nesneler arası arayüz

Kullanıcı arayüzleri:

Kullanım kolaylığı ve öğrenim zamanı esastır

Program=arayüz yaklaşımı vardır

Genel prensipler:

Veri giriş formlarının tutarlı olması

Önemli silmelerde teyit alınması

Yapılan çoğu işlem geri alınabilmesi

Hataların affedilmesi yanlış girişte kırılmama

Komut isimlerinin kısa ve basit olması

Menülerin ve diğer etkileşimli araçların standart yapıda kullanımı

Bilgi Gösterimi:

Konu çerçevesinde bilgi gösterilmeli. Kullanıcı bunaltılmamalı. Grafik ve resim kullanılmalı. Hata mesajları açıklayıcı ve anlaşılır olmalı. Değişik tür bilgiler kendi içinde sınıflandırılmalı. Rakamsal ifadelerle analog görüntü verilmeli(%89 değil).

Veri girişi

Kullanıcı hareketleri en aza indirilmeli

Gösterim ve girdi sahaları birbirinden ayrılmalı (renk uyumu önemli)

Kullanıcı uyarlamasına izin verilmeli, kullanıcı bazı özellikleri tanımlayabilmeli

Kullanılan konuyla ilgili gereksiz komutlar deaktifleştirilmeli

Bütün girdiler için yardım kolaylıkları olmalı

Kullanıcı Arayüz prototipi

Tasarım çalışması sonucunda daha önceden gereksinim çalışması sırasında hazırlanmış olan kullanıcı arayüz prototipi,ekran ve rapor tasarımları biçimine dönüşür. Ekranlar son halini alır, raporlar kesinleşir. Kullanıcıya gösterilerek onay alınır .

Tüm programının tek elden çıktığının ifade edilebilmesi açısından tüm ekranların aynı şablon üzerine oturtulması önemlidir.

Menü çubuğu Gövde (değişebilir)

Araç çubuğu Durum çubuğu

Başlangıç Tasarım Gözden Geçirme

Analiz aşamasında belirlenen gereksinimleri karşılayıp karşılanmadığının belirlenmesidir.

Sistem gereksinimlerine yardımcı olan kullanıcılar, sistem analizi yapan çözümleyiciler, sistemin kullanıcıları, tasarımcılar, yönlendirici, sekreter, sistemi geliştiricek programcılar'dan oluşan bir grup tarafından yapılır.

Ayrıntılı Tasarım Gözden Geçirme

İlk aşama tamamlandıktan sonra tasarımın teknik uygunluğunu belirlemek için yapılır. Bu çalışmada;

Çözümleyiciler , Sistem tasarımcıları, Sistem geliştiriciler, Sekreter'den oluşan bir ekip kullanılır.

Kalite Tasarım Ölçütleri

Bağlaşım (coupling):

Tasarımı oluşturan modüller arası ilişki ile ilgilidir.

Yapışıklık (cohesion):

Modülleerin iç yapısı ile ilgilidir.

Bağlaşım

Modüller arası bağlılığın ölçülmesi için kullanılan bir ölçüttür. Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır yani her bir modül olabildiğince farklı işler yapacak şekilde tasarlanmalı ve diğer modüllerle olabildiğince az etkileşimi olmalıdır.

Bağlaşımın düşük olması;

Hatanın dalgasal yayılma özelliğinin azalması, modüllerin bakım kolaylığı ,modüller arası ilişkilerde karmaşıklığın azaltılması nedenleriyle istenmektedir.

Bazı türleri vardır:

Yalın Veri Bağlaşımı

Herhangi iki modül arası iletişim yalın veriler (tamsayı,karakter,boolean vs.)aracılığla gerçekleştiriliyorsa bu iki modül yalın veri bağlaşımlıdır.

Karmaşık Veri Bağlaşımı

Herhangi iki modül arası iletişimde kullanılan parametreler karışık veri yapısı(kayıt,dizi,nesne vs.)olması durumunda tanımlanır.

Denetim Bağlaşımı

Herhangi iki modül arası iletişim parametresi olarak denetim verisi kullanılıyorsa olur.

Ortak Veri Bağlaşımı

Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül ortak veri bağlaşımlı olarak tanımlanır.

Fazla istenmeme sebepleri; ortak veri alanını izlemek zordur, ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler, ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanacak bütün modüller göz önüne alınmalıdır.

İçerik Bağlaşımı

Modüllerin iç içe tasarlanması sonucu, bir modülün başka bir modül içerisinde tanımlanmış veri alanına erişebilmesi olanaklaşır ve bu durum içerik bağlaşımına yol açar.

Yapışıklık

Bağlaşım istenmeyen bir özellikken yapışıklık istenen bir özelliktir. İkisi ters orantılıdır.

Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir ölçüttür. Modül gücü olarak da tanımlanır.

İşlevsel Yapışıklık

Tek bir iş problemine ilişkin sorunu çözen modül olarak tanımalanır. Örneğin maas_Hesapla adlı bir methodumuz modülümüz maaş hesaplama dışında bir iş yapmaz.

Sırasal Yapışaklık

Bir modülün içindeki işlemler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanıyorsa bu modüldür.

Ham_veri_kaydını_düzelt

Düzeltilmiş_ham_veri_kaydını_doğrula

Doğrulanmış_kaydı_gönder

İletişimsel Yapışıklık

Bir modülün içindeki farklı işlemler aynı girdi ya da çıktıyı kullanıyorlarsa bu modüldür.

Sicil_no_yu_al

Adres_bilgilerini_bul

Telefon_bilgisini_bul

Yordamsal Yapışıklık

Bu modüldeki işlemler arasında denetim ilişkisi bulunmaktadır. İşlemlerin birbiriyle veri ilişkisi yoktur,ancak işlem sırası önemlidir.

Ekran_görüntüsü_yaz

Giriş_kaydını_oku

Zamansal Yapışıklık

Modül içerisindeki işlemlerin belli bir zaman içerisinde uygulanması gerekiyor fakat bu işlemlerin birbirleriyle bir ilişkisi yok, yani işlem sırası önemli değilse zamansal yapışıklık vardır.

Kapıyı_Aç Kamerayı_Çalıştır

Mantıksal Yapışıklık

Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller mantıksal yapışık olarak adlandırılır. Örnek: dizilere değer atama işlemleri

Gelişigüzel yapışıklık

İşlemler arasında herhangi bir ilişki bulunmaz.

Ara_kayıt_oku

Stok_kutuğu_oku

Hata_iletisi_yaz

Yapışıklık hesaplama

Genel olarak bir modülün yapışıklık düzeyini hesaplamada bazı güçlüklerle karşılaşılabilir. Çünkü bir modüldeki işlemler hep aynı tür özellik göstermeyebilir.

Yapışıklık türü	Ölçek	Kalite
İşlevsel	1	En iyi
Sırasal	2	↑
İletişimsel	3	
Yordamsal	4	
Zamansal	5	
Mantıksal	6	
Gelişigüzel	7	En kötü

Örnek Tasarım Raporu İçeriği:

Tasarım Raporu İçeriği

Örnek Tasarım Raporu İçeriği

1. Giriş

1.1. Kapsam
1.2. Tanımlar, Kısa Adlar ve Kısaltmalar
1.3. Referanslar
1.4. Belgenin Yapısı

2. Genel Tasarım Bilgileri

2.1. Genel Sistem Tanımı
2.2. Varsayımlar ve Kısıtlamalar
2.3. Sistem Mimarisi
2.4. Dış Arabirimler

2.4.1. Kullanıcı Arabirimleri
2.4.2. Veri Arabirimleri
2.4.3. Diğer Sistemlerle Arabirimler

2.5. Veri Modeli
2.6. Testler
2.7. Performans

3. Veri Tasarımı

3.1. Tablo tanımları
3.2. Tablo- İlişki Şemaları
3.2. Veri Tanımları
3.3. Değer Kümesi Tanımları

4. Süreç Tasarımı

4.1. Genel Tasarım
4.2. Modüller

4.2.1. XXX Modülü

4.2.1.1. İşlev
4.2.1.2. Kullanıcı Arabirimi
4.2.1.3. Modül Tanımı
4.2.1.4. Modül iç Tasarımı

4.2.2. YYY Modülü

4.3. Kullanıcı Profilleri
4.4. Entegrasyon ve Test Gereksinimleri

Örnek Tasarım Raporu İçeriği

5. Ortak Alt Sistemlerin Tasarımı

5.1. Ortak Alt Sistemler
5.2. Modüller arası Ortak Veriler
5.3. Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri
5.4. Güvenlik Altsistemi
5.5. Veri Dağıtım Altsistemi
5.6. Yedekleme ve Arşivleme İşlemleri

EKLER

EK-A Rapor Görünümleri
EK-B Test Belirtimleri
EK-C Kullanıcı Kılavuzu Örneği

Yazılım Gerçekleştirm

❖ **Gerçekleştirme Fazı:**

a) Programlama Dilleri
b) Yaılım Gerçekleştirme
c) Derleyiciler
d) Kodlama Etkinliği ve Kuralları
e) Modüler Gerçekleştirme
f) Gerçekleştirmenin Belgelendirilmesi

❖ **Yazılım Geliştirme Ortamlarında bulunan unsurlar:**

- Programlama Dili
- Veri Tabanı Yönetim Sistemi
- Hazır Program Kitapçıkları(Kütüphane dosyaları)
- CASE Araçları

a) Programlama Dilleri:

Geliştirilecek uygulamaya göre kullanılabilecek programlama dili de değişmektedir.

***Uygulama alanlarının sınıflaması:**

-Veri işleme yoğunlamalar

-Hesaplama yoğunluklu uygulamalar

-Süreç ağırlıklı uygulamalar

-Sistem programlamaya yönelik uygulamalar

-Yapay (zeka-akıl) uygulamaları biçiminde verilmektedir.

- **Veritabanı Yaklaşımının Geleneksel Kütük Kavramından Ayıran Özellikleri:**

1-Veri sözlüğü. 2- Veri soyutlama. 3- Program-Veri ve program -işlem bağımsızlığı

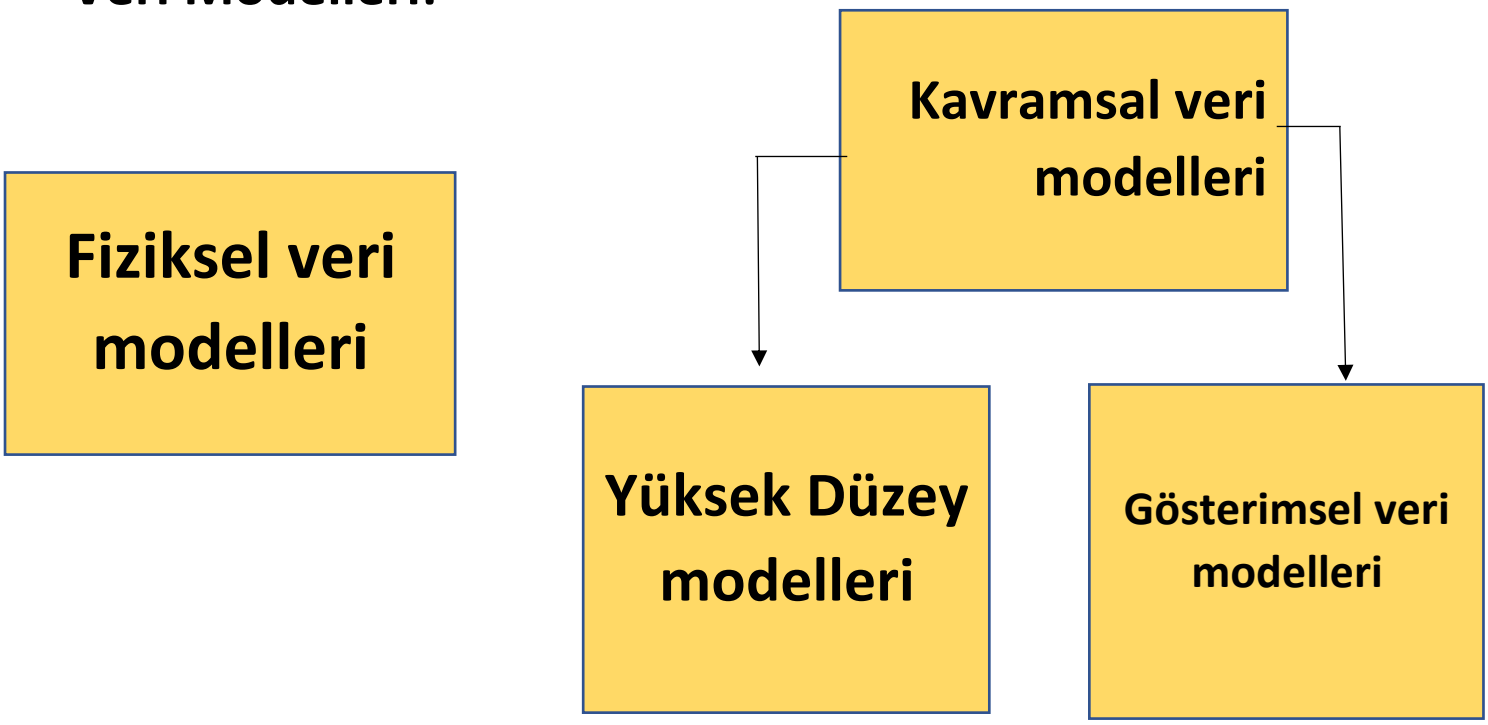
4- birden çok kullanıcı desteği 5- Verinin birden fazla işlem arasında paylaşımı

- **Veri Soyutlama:**

Veri tabanı yaklaşımının bir temel karakteristiği kullanıcıdan verinin saklanması Konusundaki detayları gizleyerek veri soyutlamasını sağlamasıdır. Bu soyutlama Sağlayan ana araç veri modelidir.

- **Program-veri ve program-işlem bağımsızlığı:**
  -Geleneksel dosya işleme yönteminde,veri dosyalarının yapısı bu Dosyalara erişim programları içine görmülüştür.
  -Tersine VTYS erişim programları veri dosyalarından bağımsız olarak Tasarlanmışlardır. VTYS de saklanan veri dosyalarının yapısı erişim Programlarından ayrı olarak katalogda tutulduğundan program ve veri bağımsızlığı sağlar.

Veri Modelleri:



❖ Şemalar:

- Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önem

Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlanılır.

Veri tabanı şenası,tasarım sırasında belirtilir ve sıkça değişmesi beklenmez.

- Pek çok veri modeli şemaları,diyagramlar halinde göstermek için belli gösterim Biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek Örneğini göstermez.

❖ CASE Araç ve Ortamları:

- CASE Araçları, yazılım üretiminin hemen her aşamasında (Planlama – Çözğmleme – Tasarım – Gerçekleştirme – sınama) üretilen bilgi ya da belgelerin bilgisayar ortamında Saklanmasını, bu yolla kolay erişilebilir ve yönetilebilir olmasını olanaklı kalır.

❖ Kodlama stili:

- Yazılım kod stili konusunda herhangi bir kabul edilmiş standart bulunmamaktadır.
- Bu konuda, yazılım geliştirilen ekiplerin, kodlama aşamasına başkamadan kodların Düzeni konusundaki standartlarını ya da kurallarını geliştirmeleri ve bu kuralların Uygulamaya geçmesini sağlamaları önerilmektedir.

❖ McCabe Karmaşıklık Ölçütü Hesaplama

Ana program ve alt programlar çizge biçimine dönüştürüldükten sonra programın McCabe karmaşıklık öiçütü V(G) aşağıdaki biçimde hesaplanır.

$$V(G) = k - d + 2p$$

K : kenar sayısı

D : düğüm sayısı

P : bileşen sayısı

❖ Gerçekleştirme Öznitelikleri:

-Etkinlik - Belirgin Arayüz

-Aykırı Durumların Kotarılması - Taşınabilirlik

-Çok düzeyli koruma

-Hata Ayıklama -Gözden geçirme

-Atık/Çöp toplama -Belgelendirme

-Soyutlama -Bilgi Gizleme -Etiketleme -Güvenlik -Basitlik

YAZILIM DOĞRULAMA VE GEÇERLEME

-Doğrulama

Doğru ürünümü üretiyoruz?

=Ürünü kullanacak kişilerin isteklerinin karşılanıp karşılanmadığına dair etkinliklerden oluşur

-Gerçekleme

Ürünü doğru olarak mı üretiyoruz?

Ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinden oluşur.

IEEE 1028 standardı; hem kritik hem de kritik olmayan yazılımlar için geçerli olan gözden geçirme ve denetleme için tanımlar ve yeknesak prosedürler sağlanır. Süreçler, yönetim incelemesi, teknik inceleme, yazılım incelemesi, izlenecek yol ve denetimi içerir. Tanımlanan süreçler, yazılım yaşam döngüsü boyunca uygulanabilir.

SVVP (Software Verification and Validation Plan)

Yazılım proje yönetimi, yazılım testi ve yazılım mühendisliğinde doğrulama ve onaylama (**V&V**), bir yazılım sisteminin amaçlanan amacını yerine getirmesi için spesifikasyonları ve **gereksinimleri** karşılayıp karşılamadığını kontrol etme sürecidir . Yazılım kalite kontrolü olarak da adlandırılabilir.

Küçükten büyüğe sınama kavramlar

1-Birim sınaması Bağlı oldukları diğer sistem unsurlarından bütünüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.

2-Alt-Sistem sınama modüllerin bütünleşmesi ile oraya çıkar. Bu düzeyde en çok hata ara yüzlerde bulunmaktadır.

3-Sistem Sınaması Üst düzeyde bileşenlerin sistem ile olan etkileşimlerinde çıkacak hataları arar.

4-Kabul sınama sistemin çalıştırılmadan önce ki son sınamasıdır. Artık yapay veriler yerine gerçek veriler kullanılır.

SINAMA YÖNTEMLERİ

Her mühendislik ürünü, iki yoldan biri ile sınanır: Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün (kara kutu – Black box) veya iç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanması (beyaz kutu – White box).

Kara kutu sınamasında sisteme, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır.

Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır.

Bütün bağımsız yollar en az bir kere sınanır, bütün döngüler sınır değerlerinde sınanır ve iç veri yapıları denenir.

Not: Kara kutu sınamasında kod sınamaları olmazken beyaz kutu sınamalarında mantıksal karar yapılarına kadar sınama yapılır. Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları, aslında çok sıkça çalıştırılıyor olacaktır. Yazım hataları rastgele olarak dağılır. Bunlardan bazılarını derleyiciler bulur, bazıları da bulunmadan kalır.

TEMEL YOLLAR SINAMASI

Bağımsız yolların saptanması için program ilk önce graf (çizgisel) biçime çevrilir. Bunu yapmak için ise, program iş akış şemaları oldukça iyi bir başlangıç noktasıdır.

SINAMA VE BÜTÜNLEŞTİRME STRATEJİLERİ

Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya koyar. Yukarıdan aşağı bütünleştirmede önce en üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir.

Alt bileşenler ise bu stratejiye göre henüz hazırlanmış olmadıklarından bunların yerine üst bileşenin sınaması için kullanılmak üzere koçan programlar yazılır. Koçanlar, bir alt bileşenin, üst bileşen ile arayüzünü temin eden fakat işlevsel olarak bir şey yapmayan boş çerçeve programlardır.

YUKARIDAN AŞAĞI SINAMA ve BÜTÜNLEŞTİRME

1-Düzey Öncelikli Bütünleştirme

En üst düzeyden başlanır, öncelikle aynı düzeylerdeki birimler bütünleştirilir.

2-Derinlik Öncelikli Bütünleştirme

En üst düzeyden başlanır. Birim şemasında bulunan her dal soldan sağa olmak üzere ele alınır. Bir dala ilişkin bütünleştirme bitirildiğinde diğer dalın bütünleştirilmesi başlar.

AŞAĞIDAN YUKARIYA SINAMA ve BÜTÜNLEŞTİRME

Önce en al düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınanması gerektiğinde bu üst bileşen, bir ‘sürücü’ ile temsil edilir, koçan ile aynı özelliklere sahiptir.

SINAMA PLANLAMASI

Sınama yönteminde yer alması gereken başlıklar: Sınama konusu, sınama etkinlikleri ve zamanlama, temel sınama teknikleri, destek etkinlikler, kaynaklar ve sorumluluklar, personel ve eğitim gereksinimleri, sınama yaklaşımı, riskler ve çözümler, onaylar.

Sınama ayrıntılarında yer alması gereken başlıklar: sınanacak sistemler, girdiler ve çıktılar, sınamaya başlanma koşulları, girdilerin hazır olması, ortam koşulları, kaynak koşulları, sınama tamamlama kıstası, sınama geçme-kalma kıstası, sınama askıya alınma kıstası ve sürdürme gerekleri, sınama sonuçları.

Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmasıdır.

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

Planlama aşamasında genel sınama planı oluşturulur. Söz konusu plan tüm sınama etkinliklerini çok genel hatlarıyla tanımlar ve sınama planlamasında verilen bilgileri içerir.

Çözümleme aşamasında, sistemler ve alt sistemler ortaya çıkarılır ve sınama planı alt sistemler bazında ayrıntılandırılır.

Tasarım aşamasında tüm yazılım modüllerinin ortaya çıkarıldığı aşamadır. Bu aşamanın başlangıcında yazılım modülleri için sınama planı detaylandırılır ve sınama belirtimleri hazırlanır. Söz konusu belirtimler, kullanıcı kılavuzları ile birlikte eğitim için temel bilgileri oluşturur.

Gerçekleştirim aşamasında teknik sınamalar yapılır, sınama raporları hazırlanır ve kullanıcı sınayıcıları eğitilir.

Bakım aşamasından hemen sonra kullanıcı sınamaları yapılarak sınama raporları hazırlanır.

Sınama sırasında bulunan her hata için değişiklik kontrol sistemine yazılım değişiklik isteği türünde bir kayıt girilir.

Onulmaz Hatalar: BT projesinin gidişini bir ya da birden gazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

Küçük Hatalar: Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.

Şekilsel Hatalar: Heceleme hatası gibi önemsiz hatalardır.

PL/SQL : Oracle tarafından geliştirilen veritabanı sistemleri özel bir dildir. Yapısal dillere ait özelliklerin SQL’e eklenmesiyle ortaya çıkmış bir dildir. SQL’den farkı ise **SQL**, verileri seçmek ve işlemek için veri odaklı bir dil olmakla birlikte, **PL SQL**, uygulamalar oluşturmak için kullanılan bir yordam dalıdır. **SQL**, bir defada bir deyimi yürütürken, **PL**' de **SQL** kod bloğu yürütülebilir. **SQL**, **PL SQL** 'in usule uygun olarak bildirimsel niteliktedir.

11.HAFTA

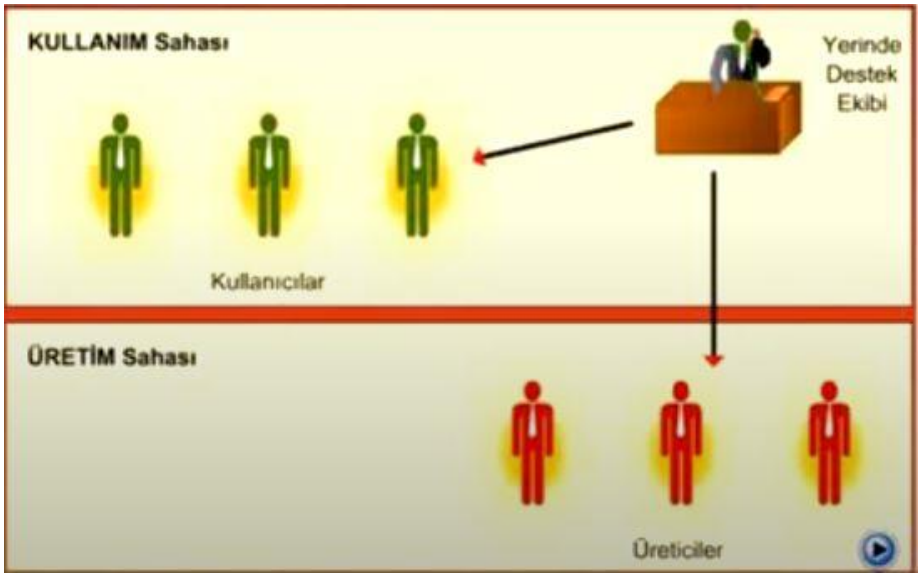
Sınama işlemleri bitirilen yazılımın kullanıcı alanında yüklenmesi ve uygulamanın başlatılması gerekmektedir.
yazılım kullanıma geçtikten sonra yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan bakım aşaması başlar.
izleye kesimlerde kurulum ve bakım aşamasında yapılması gerekenler açıklanmaktadır.
bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı dikkate alınmıştır.

Kurulum
sınanmış yazılımların kullanıcı sahasına aktarılması ve yazılımın gerçek yaşamda uygulanmasının başlatılması için yapılan işlemler kurulum işlemleri olarak tanımlanır.

- donanım kurulumu
- sistem yazılımları kurulumu
- veritabanı kurulumu
- uygulama yazılımları kurulumu
- eğitim
- yerinde destek

Yerinde Destek Organizasyonu

- yerinde destek ekibi kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir. temel görevleri;
- kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak, giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasında uygulama yazılım destek ekibine iletmek
- Kullanıcıya iş başında uygulama eğitimi vermek
- kullanıcı sınama günlükleri toplamak
- Yapılan tüm işlemleri konfigürasyon veritabanına kaydetmek biçimindedir



Bakım Süreç Modelleri;

- sorun tanımlama süreci
- çözümleme süreci
- tasarım süreci
- gerçekleştirim süreci
- Sistem sınama süreci
- kabul sınaması süreci
- kurulum süreci başlıkları altında incelenir.

Sorun tanımlama süreci

Bakım tanımı: bakım işletimi alınan yazılımın sağlıklı olarak çalışması ve ayakta kalabilmesi için yapılması gereken çalışmaların bütünü olarak tanımlanır. Uygulamada çalışan bir yazılımın 3 tür bakım gereksinimi bulunur.

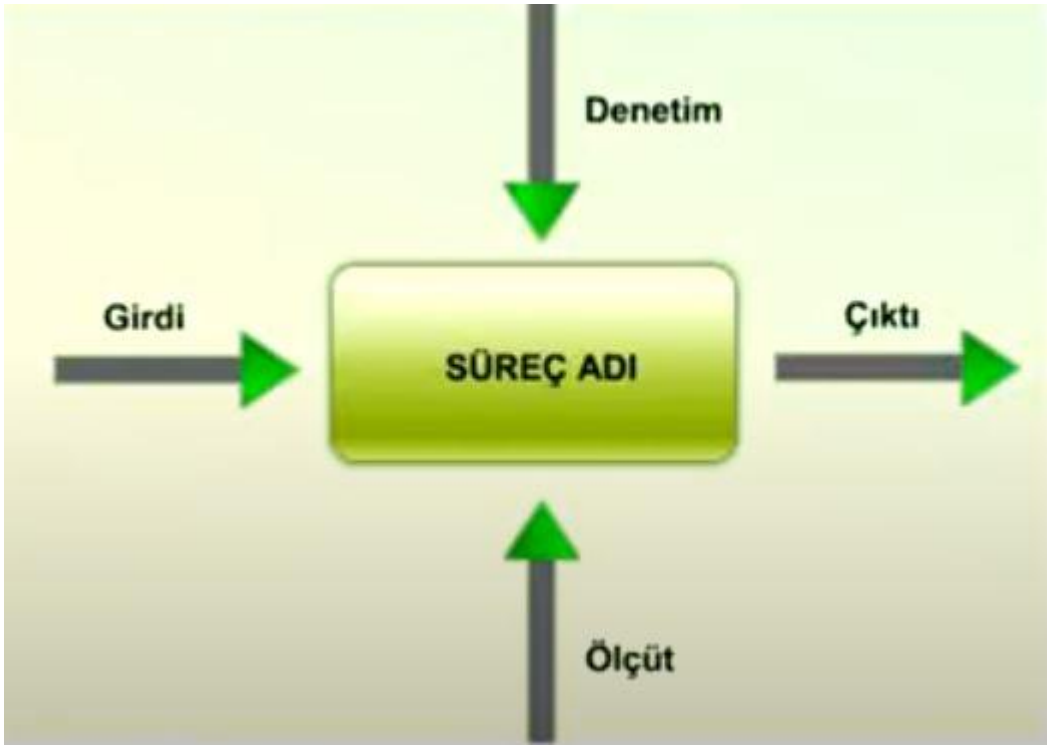
- Düzeltici bakım
- uyarlayıcı bakım
- en iyileyici bakım

<u>Düzeltici bakım</u>; yazılım sınanması teorik olarak tamamen yapılabilirse de pratikte pek sağlanamaz bu nedenle yazılımda her an bir hata ile karşılaşabiliriz hataların düzeltilmesi için de düzeltici bakım devreye girer.

Uyarlayıcı bakım; uygulama yazılımları işletme ya da kuruluşların günlük yaşamda yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan araçlardır. işletme ya kuruluşlarda yaşanan değişikliklerin o kuruluşun işlerini bilgisayar yardımı ile yapmalarını sağlayan uygulama yazılımlarına da yansıtılması gerekir. bu yansıtma işlemi uyarlayıcı bakım olarak tanımlanır.

En iyileyici bakım; zaman zaman uygulama yazılımlarının çalışma performanslarının iyileştirilmesi amacıyla yapılan çalışmalar en iyileyici bakım olarak tanımlanır.

Bakım süreç modeli:
IEEE 1219 standardı tarafından önerilen bakım süreç modeli şekilde belirtilen şablon kullanarak bakım süreçlerini tanımlamaktadır.



Bakım süreç modelinin süreçleri:

- Sorun tanımlama/sınıflandırma
- Çözümleme
- Tasarım
- Gerçekleştirim
- Sistem sınama
- Kabul sınama
- Kurulum

Bakım süreci yazılım yaşam döngüsü çekirdek adımlarının bir anlamda yinelenmesinden oluşmaktadır.

Sorun tanımlama süreci;

Girdi; temel girdi bakım isteğidir.
bakım istekleri:

- sistemde beklenen ve yeni düzenlemelere ilişkin değişiklikler
- Yeni işlev istekleri
- Yazılımda bulunan yanlışların düzeltilme istekleri
- performans artırımına ilişkin istekler
- yeni iş yapma türlerine ilişkin istekler
- Teknolojinin zorlaması sonucu oluşan istekler örnek verilebilir

<u>İşlem/Süreç</u>:
Yazılım bakım isteği oluştuğunda yapılması gerekenler;

- Değişiklik isteğini bir tanım numarası verilmeli
- değişiklik türünün sınıflandırılması
- değişik isteğinin kabul red ya da daha ayrıntılı incelenmesi yönünde karar alınması
- değişiklik ile ilgili zaman boyut işgücü kestirme yapılması
- değişiklik isteğinin diğerleri ile birlikte zaman ve iş planına kaydedilmesi

Birçoğunda değişiklik isteyen kişi, kullanıcı temsilcileri, yazılım mühendisleri, konu uzmanlarıdır.

<u>Denetim</u>: sorun tanımlama aşamasında değişiklik isteğinin daha önceden yapılıp yapılmadığını denetlemeli ve tek olduğu belirlenmelidir. bu amaçla daha önceki değişiklik istekleri taranır.

<u>Çıktı</u>:

Temel çıktısı doğrulanmış, geçerlenmiş ve karar verilmiş bakım isteğidir.
Veritabanında saklanan isteğin ayrıntıları;

- Sorun ya da yeni gereksinim tanımı
- sorun ya da yeni gereksinim değerlendirilmesi
- Başlangıç önceliği
- geçerleme verisi
- Başlangıç kaynak gereksinimi
- mevcut ve gelecekte kullanıcılar üzerinde etkileri
- yararlı ve aksak yönleri

Ölçüt: oğlum tanımlama sırasında kullanılabilecek ölçütler;

- Bakım isteklerinde kabul edilmeyen madde sayısı
- gelen bakım istekleri sayısı
- sorun geçerleme için harcanan kaynak ve zaman biçimindedir.

Çözümleme süreci; veritabanında saklanmış ve geçerlenmiş bakım isteğini girdi olarak alır, projeye ilişkin bilgi ve belgeleri kullanarak söz konusu isteğinin yerine getirilmesi için gerekli genel planı yapar.
Resim 5

Girdi;

- geçerlenmiş bakım isteği
- başlangıç kaynak gereksinimleri ve diğer veriler
- mevcut proje ya da sistem bilgi ve verileri biçimindedir.

İşlem/Süreç: olurluk aşaması ve ayrıntılı çözümleme aşamasıdır.

Bu süreç, değişikliğin etkisi,

- Prototipleme içeren seçenek çözümler
- dönüştürme gereksinimlerinin çözümleri
- güvenlik ve emniyet zorunlulukları
- İnsan faktörleri
- kısa ve uzun verimli faaliyetler
- değişikliği yapmanın yararları içerir.

GEREKSİNİM VERİ TOPLAMA

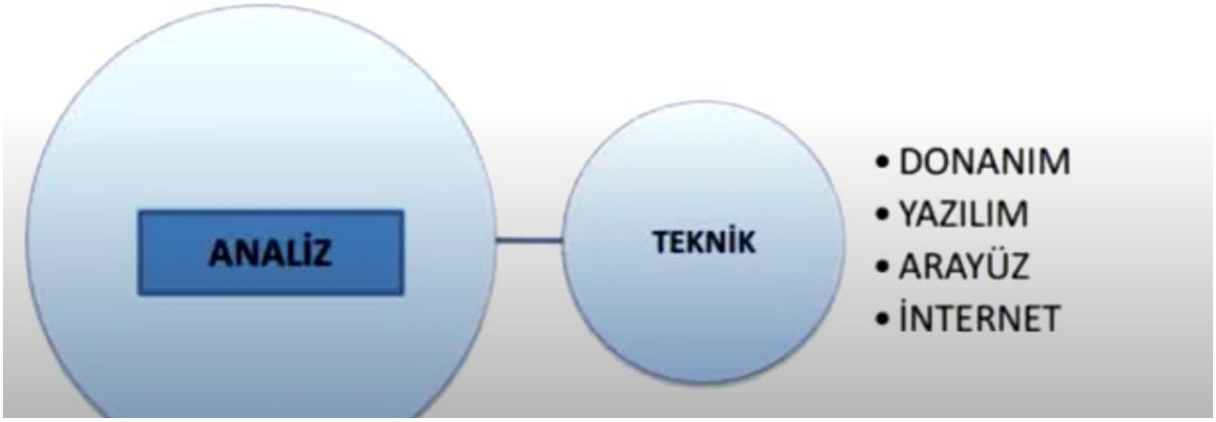
-Yazılım mühendisliği projelerinin geliştirilmesi sürecinde en önemli unsurlardan biriside mevcut sistemin incelenmesidir. Genel olarak müşteri ile görüşmeler ve mevcut sistemin incelenmesi sonucunda gereksinim analizi dökümanı üretilir

Mevcut sistemin incelenmesi sırasında kullanılabilecek temel yöntemler aşağıdaki gibi özetlenebilir;

1-Sorma

2- Psikolojik Türetme Teknikleri

3-İstatiksel Teknikler



NOT: Yaptığımız donanım donanım ürünlerini karşılıyor mu ihtiyaç var mı ? İşletim sistemleri? Lisanslar mevcut mu ? Vüris programları ? Daha önce kullanılmış uygulama var mı (programla combined uygulama var mı)? Mobil mi Masaüstümü ?Ekran tasarımı nasıl olsun ? Sistemin yükünü karşılayacak ağ yapısı var mı ? Güncelleme karşı internet üzerinden müdahale edilebilir mi ? vs gibi sorular kullanıcıya sorular

KOD BANKASI PROJESİ VERİ SÖZLÜĞÜ ÖRNEĞİ

-Kod Bankası Projesi en önde gelen hedefi kod aramalarında en çok ziyaret edilen ; kodlara hızlı erişim sağlayan ve güvenilir içeriği ile yazılım sektöründeki kullanıcıların vazgeçemeyeceği bir web kaynak sayfasını oluşturmaktadır.(Programcıya örnek ve yol gösterici olabilir)

İşlev Gereksinimleri

-Müşteri kullanıcı adı ve şifre , isim ,e-mail adresi ile kullanıcıların oturuma girmesini ister

-Kullanıcıların , farklı programlama dillerinde kod yükleyebilmesini ister

-Kullanıcıların yükledikleri kodları çalıştırabilmelerini ister

-Kullanıcıların veri tabanına kolay ve hızlı bir şekilde tarama yapabilmelerini ister

KOD BANKASI PROJESİ VERİ SÖZLÜĞÜ örneği (sitede paylaşılanlar)

Dosya Adı	Değişken Adı	Veri Tipi	Uzunluk	Açıklama
Users	user_id	int	4	Kullanıcı Numarası (ID)
	nick	varchar	10	Kullanıcı Adı
	fullname	varchar	20	Kullanıcı Adı Soyadı
	email	varchar	20	Kullanıcı E-Mail Adresi
	password	varchar	8	Kullanıcı Şifresi
	accesslevel	int	1	Kullanıcı Yetkilendirme Seviyesi (0..9)
Codes	code_id	int	6	Kod Numarası (ID)
	language	varchar	15	Programlama Dil İsmi
	code_category	varchar	15	Kod Kategorisi
	add_date	datetime		Kod Ekleme Tarihi
	score	int	2	Kod Puanı (0-10)
	search_word	varchar	30	Kod Arama Kelimeleri
Forum	comments	longtext		Forum Yorumları
	forum_category	varchar	15	Forum içerisindeki dil kategorileri
	forum_date	datetime		Foruma bilgi giriş tarihi

Not: Burdaki asıl amaç github gibi siteleri anlatıyor demek isteği programcıya destek siteleri

! HER PROJE FİKİRLE BAŞLAR

Sistem Tasarım Aşaması

- Yazılım Geliştirme Ortamları
- Kodlama Stili
- Program Karmaşıklığı
- Olağan Dışı Durum Çözümleme
- Kod Gözden Geçirme

Sistem Gerçekleştirim Aşaması

- Sınama Kavramları
- Doğrulama ve Geçerleme Yaşam Döngüsü
- Sınama Yöntemleri
- .Sınama ve Bütünleştirme Stratejileri
- Sınama Planlaması
- Sınama Belirtimleri
- Yaşam Döngüsü Boyunca Sınama Etkinlikleri

Doğrulama ve Geçerleme Aşaması

- Kurulum
- Yerinde Destek Organizasyonu
- Yazılım Bakımı

Bakım ve Destek Aşaması

SONUÇ **KAYNAKLAR**