Bölüm 9

Yazılımın Evrimi (Software Evolution)

İçerik

- Evrim Süreçleri
  - Yazılım sistemleri için değişim süreçleri
- Program Evrim Dinamikleri
  - Yazılım evrimini anlama
- Yazılımın Bakımı
  - Çalışan yazılım sistemlerinde değişiklik yapmak
- Eski Sistemlerin Yönetimi
  - Yazılım değişimi hakkında karar vermek

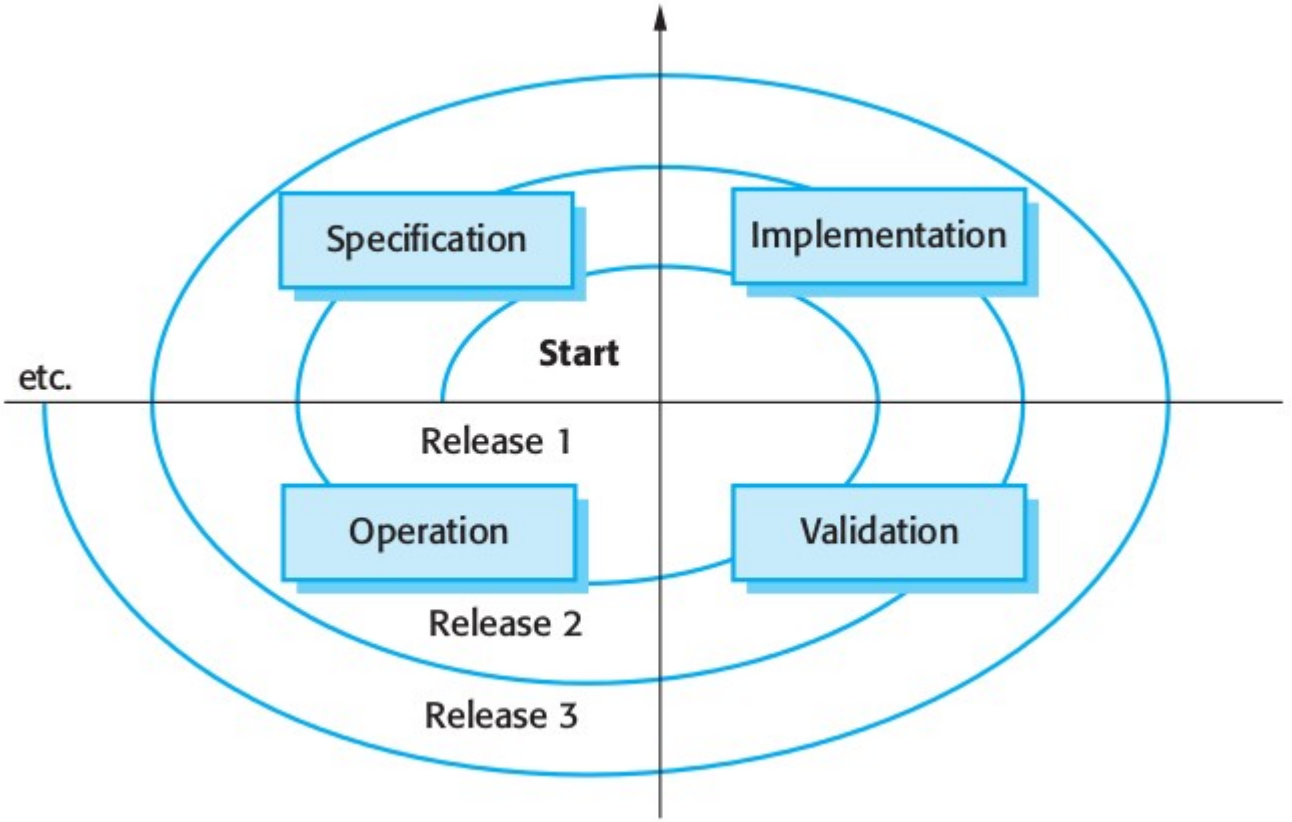
Yazılımda Değişiklik

- Yazılımda değişiklik kaçınılmazdır
  - Yazılım kullanılırken yeni gereksinimler ortaya çıkar
  - İş çevresi değişir
  - Hatalar düzeltilmek zorundadır
  - Yeni bilgisayarlar ve ekipmanlar sisteme eklenir
  - Sistemin performansı veya güvenilirliği geliştirilebilir
- Tüm kurumlarda mevcut yazılıma değişiklikler yapılması ve değişikliklerin yönetimi önemli bir problemdir

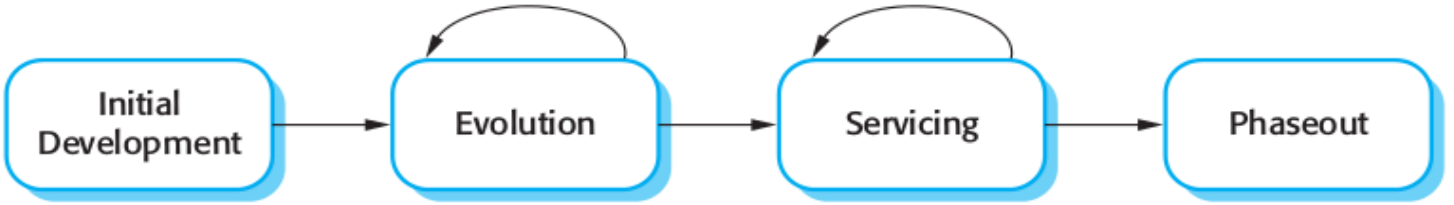
Evrimin Önemi

- Kurumlar yazılım sistemlerine çok büyük yatırımlar yaparlar – yazılımlar kritik iş varlığıdır
- Bu varlıkların değerini korumak için değişiklik ve güncelleme şarttır
- Büyük şirketlerde yazılım bütçesinin büyük bir kısmı yazılımın geliştirilmesinden çok bakımı ve evrimine ayrılır

Geliştirme ve Evrimin Spiral Modeli



Evrim ve Bakım



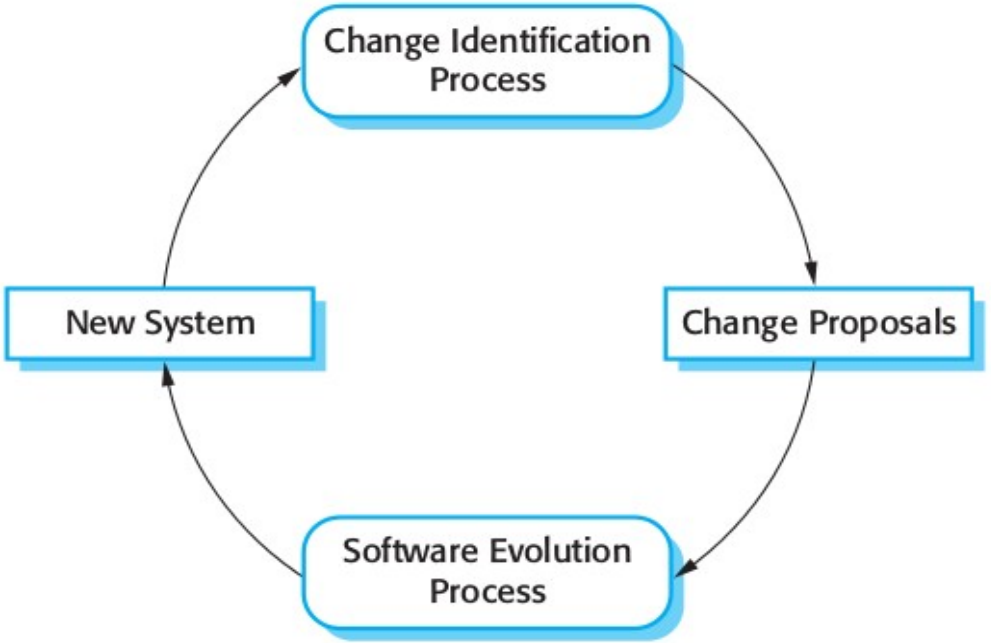
Evrim ve Bakım

- Evrim
  - Yazılım gerçek hayatta kullanılmaya başlanmıştır; yeni gereksinimlerin ortaya çıkması ve bunların sistemde gerçeklenmesi ile evrimleşmektedir
- Bakım
  - Bu aşamada, yazılım kullanılmaya devam etmektedir ancak yeni gereksinimler alınmaz ve sisteme yeni fonksiyonellikler eklenmez. Sadece sistemin çalışmasını sürdürmesi için gereken hata düzeltme işlemleri yapılır
- Phase-out
  - Sistem kullanımdadır ancak üzerinde herhangi bir değişiklik yapılmamaktadır

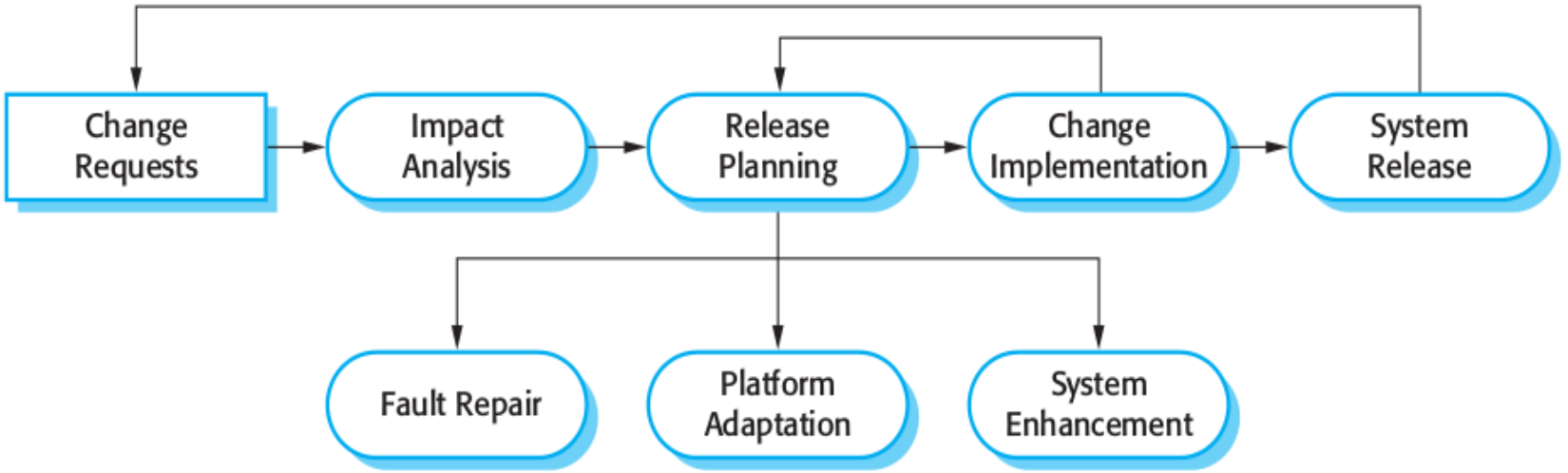
Evrim Süreci

- Yazılım evrim süreci şunlara bağlıdır:
  - Yazılımın tipi
  - Kullanılan geliştirme süreçleri
  - İnsanların yetenekleri ve deneyimleri
- Değişim önerileri sistemin evrimi için etmendir
- Değişim tanımı ve evrim, sistemin yaşam döngüsü boyunca devam eder

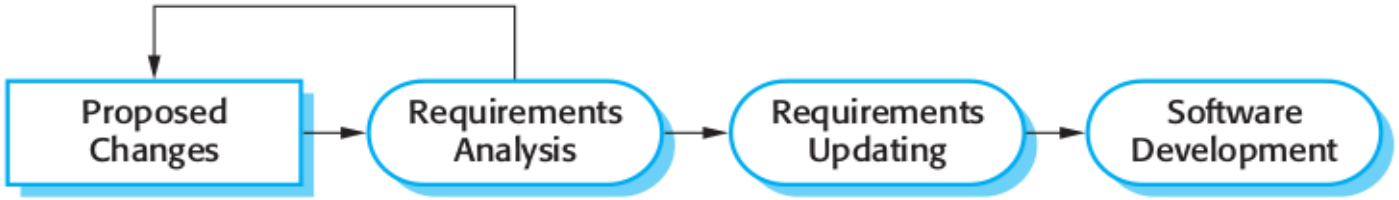
Değişim Tanımı ve Evrim Süreci



Yazılım Evrim Süreci



Değişiklikleri Gerçekleştirme



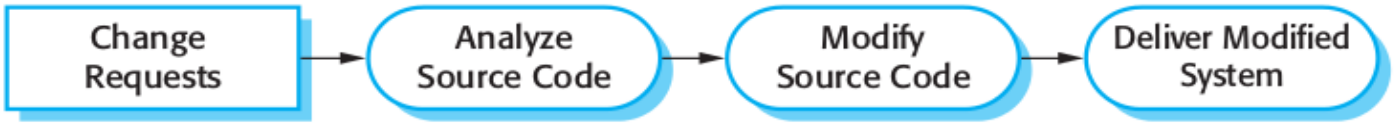
Değişiklikleri Gerçekleştirme

- Sistemde değişikliklerin tasarlandığı, gerçekleştirildiği ve test edildiği geliştirme süreci adımıdır
- Eğer sistemi geliştirenlerden başkaları değişiklikleri gerçekleştirmekten sorumlu ise, ilk olarak sistemin anlaşılması gerekir ve bu çok kritik bir adımdır
- Programı anlama sürecinde, programın nasıl yapılandırıldığı, işlevselliğini nasıl gerçekleştirdiği ve önerilen değişikliğin programı nasıl etkileyeceği anlaşılmak zorundadır

Acil Değişiklik Talepleri

- Acil değişikliklerin tüm yazılım geliştirme adımlarının üzerinden geçilmeden gerçekleştirilmesi gerekebilir
  - Yazılımın normal çalışmasını sürdürebilmesi için bir sistem hatasının düzeltilmesi gerekiyorsa
  - Eğer sistemin çevresindeki bir değişiklik beklenmedik etkilere yol açtıysa (örneğin işletim sistemi güncellemesi)
  - Eğer iş dünyasında çok hızlı hareket edilmesi gereken bir değişim olduysa (rakip ürünün yeni sürümünün çıkması)

Acil Durum Düzeltme Süreci



Çevik Yöntemler ve Evrim

- Çevik yöntemler artırımlı geliştirimi temel aldığı için geliştirme aşamasından evrime geçiş görünür değildir
  - Evrim, sık sık çıkarılan sistem sürümleri üzerinden geliştirmeye devam etmektir
- Bir sistemde değişiklikler yapıldığında otomatik regression testleri özellikle önemlidir
- Değişiklikler, ek kullanıcı hikayeleri şeklinde ifade edilebilir

Devir-Teslim Sorunları

- Geliştirme takımı çevik yöntemler kullandı fakat evrim takımı çevik yöntemlere aşina değil, plan-tabanlı yaklaşım kullanıyor
  - Evrim takımı detaylı dokumantasyon bekleyebilir ancak çevik süreçte bu üretilmemektedir
- Geliştirme takımı plan-tabanlı yaklaşım kullandı ancak evrim takımı çevik yöntemleri tercih ediyor
  - Evrim takımı taslaktan başlamak ve otomatik testler geliştirmek zorunda kalabilir; sistem kodu çevik yöntemlerden beklendiği şekilde refactor edilmemiş ve basitleştirilmemiş olabilir

Program Evrim Dinamikleri

- Program evrim dinamikleri sistem değiştirme süreci çalışmasıdır
- Birçok amprik çalışma sonrası Lehman ve Belady her sistem evrildikçe uygulanabilen bazı kanunlar belirlemişlerdir
- Kanunlardan daha ziyade mantıklı gözlemler mevcuttur. Büyük kurumlar tarafından geliştirilen büyük sistemlere uygulanabilirler
  - Diğer türde yazılım sistemlerine uygulanabilirlikleri tartışılır

Değişim Kaçınılmazdır

- Sistemler geliştirilirken sistem gereksinimleri genellikle değişmektedir çünkü çevresel faktörler değişim gösterir. Dolayısıyla oluşan sistem gereksinimleri karşılamayacaktır
- Sistemler kendi çevreleri ile sıkı bir ilişki içindedir. Bir sistem bir ortama yüklendiğinde o ortamı ve dolayısıyla sistem gereksinimlerini değiştirir
- Sistemler bir ortamdaki kullanışlılıklarını korumak için değişmek zorundadır

Lehman Kanunları

Kanun	Tanımı
Sürekli değişim	Gerçek dünyada kullanılan bir program değişmek zorundadır ya da o ortamda çok az kullanılmaya başlar.
Artan karmaşıklık	Program evrildikçe yapısı daha karmaşıklaşmaya başlar. Program yapısını korumak veya basitleştirmek için ekstra kaynak ayrırmak gerekebilir.
Büyük programların evrimi	Programın evrimi bir kendi kendini düzenleme sürecidir. Sistem boyutu, sürümler arasında geçen zaman ve bildirilen hatalar gibi sistem özellikler her sistem sürümü için aşağı yukarı aynıdır.
Kurumsal istikrar	Programın yaşam süresi boyunca geliştirme oranı neredeyse sabit kalır ve sistem geliştirme için ayrılan kaynaklardan bağımsızdır.

Lehman Kanunları

Kanun	Tanımı
Benzerliklerin korunması	Sistemin yaşam süresi boyunca her sürümdeki artımlı değişim neredeyse sabittir.
Gelişimi sürdürme	Sistemlerin fonksiyonellikleri kullanıcı taleplerini karşılamak üzere sürekli artmak durumundadır.
Kalitede azalma	Değişimleri yansıtmadıkları sürece sistemlerin kalitesi düşecektir.
Geri bildirim sistemi	Evrim süreci, ürün üzerinde önemli gelişmeler kaydedebilmek için çok etmenli, çok döngülü geri bildirim sistemleriyle bütünlük sağlar.

Lehman Kanunlarının Uygulanabilirliği

- Lehman kanunları, büyük ölçekli kurumlar tarafından geliştirilen büyük ve kişiye / kuruma özel yazılımlar için uygulanabilir görünmektedir
  - 2000'lerin başında FEAST projesinde uygulanması ile kanıtlanmıştır
- Şu tip sistemler için nasıl değiştirilebilecekleri belirsizdir:
  - Ambalajlı yazılım ürünleri
  - Birçok COTS bileşenine sahip sistemler
  - Küçük ölçekli kurumlar
  - Orta ölçekli sistemler

Önemli Noktalar

- Yazılım geliştirme ve evrim spiral model kullanılarak belirtilebilecek, birleşik ve tekrarlayan süreçler olarak düşünülebilir
- Özel sistemler için, yazılımın bakım maliyeti genellikle geliştirme maliyetini aşar
- Yazılım evrim süreci değişiklik istekleri ile başlar ve değişim etki analizi, süreç planlama ve değişikliğin uygulanması gibi işler içerir
- Değişikliğin sürekli olduğu gibi Lehman kanunları, sistem evrimi üzerine yapılan uzun süreli çalışmalardan elde edilen kavramlardır

Yazılımın Evrimi

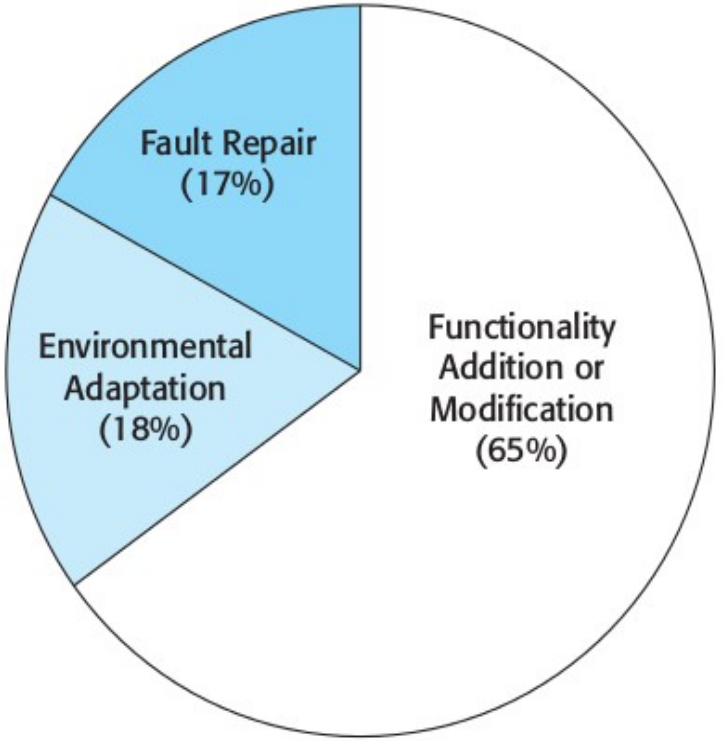
Yazılımın Bakımı

- Program kullanılmaya başladıktan sonra değişiklik yapılmasıdır
- “Bakım” terimi genellikle özel yazılımların değişiklikleri için kullanılır. Genel amaçlı yazılımlar yeni sürümler çıkararak evrimleşirler
- Bakım, normalde sistem mimarisinde yapılan büyük değişimleri kapsamaz
- Değişiklikler mevcut bileşenleri değiştirme ve sisteme yeni bileşenler ekleme yoluyla gerçekleştirilir

Bakım Tipleri

- Yazılım hatalarını gidermek için bakım
  - Gereksinimleri karşılama yolundaki bozuklukları düzeltme amacıyla değişiklik yapma
- Yazılımı başka bir çalışma ortamına uyarlamak için bakım
  - Sistemin mevcut halinde, başka ortamlarda çalışabilmesi için değişiklikler yapmak
- Sistemin işlevselliğini değiştirmek veya yeni işlevsellik eklemek için bakım
  - Yeni gereksinimlere cevap verebilmek için sistemde değişiklikler yapmak

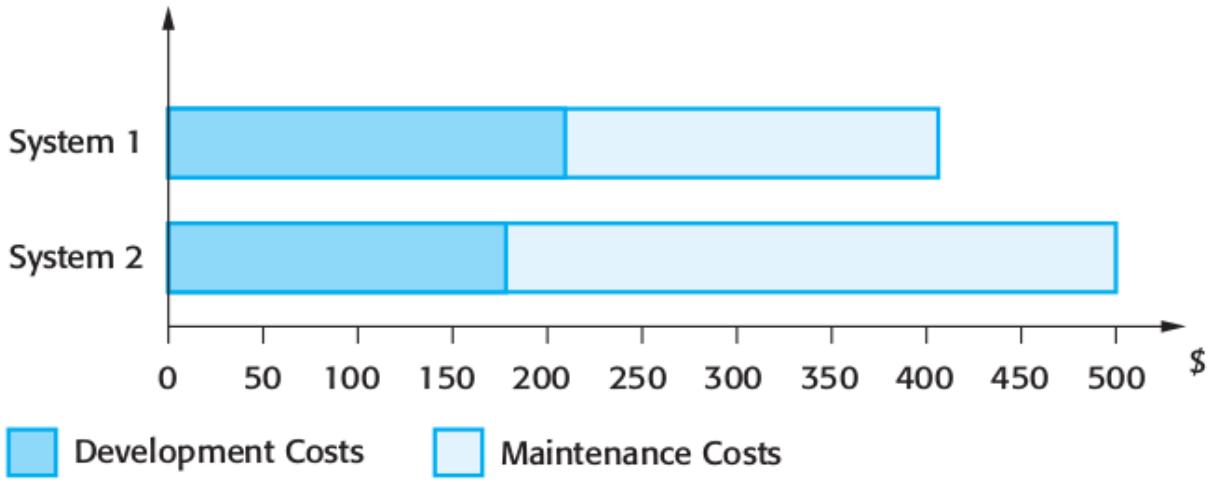
Bakım Emek Dağılımı



Bakım Maliyeti

- Genellikle geliştirme maliyetinden fazladır (uygulamaya göre 2 ila 100 kat)
- Teknik ve teknik olmayan faktörlerden etkilenir
- Yazılıma bakım yapıldıkça artar. Çünkü bakım yazılımın yapısını bozar ve bu da daha sonra yapılacak bakımı zorlaştırır
- Yaşlanmış yazılım desteği yüksek maliyetli olabilir (eski programlama dilleri, derleyiciler, ...)

Geliştirme ve Bakım Maliyetleri



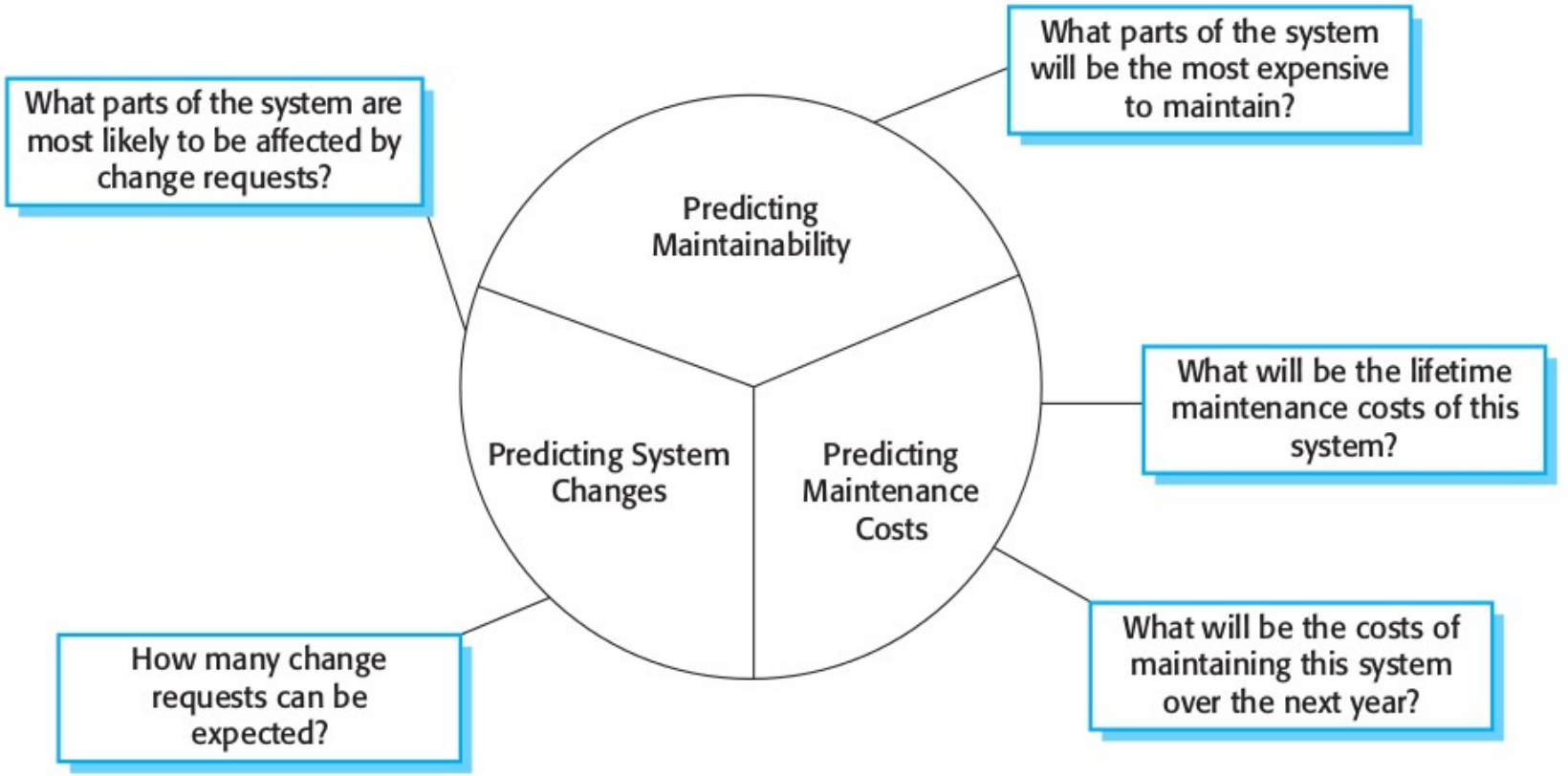
Bakım Maliyetini Etkileyen Faktörler

- Takım istikrarı
  - Eğer bakım bir süre boyunca aynı takım tarafından yürütülüyorsa maliyet azalır
- Sözleşme sorumlulukları
  - Sistem geliştiriciler bakım yapmak için bir sözleşme imzalamadılarsa, gelecekte yapılacak değişiklikler için istekli olmazlar
- Çalışanların yetenekleri
  - Bakım ekibi genellikle deneyimsiz ve kısıtlı alan bilgisine sahip kişilerdir
- Programın yaşı ve yapısı
  - Program yaşlandıkça yapısı bozulur ve daha zor anlaşılır ve daha zor değiştirilebilir hale gelir

Bakım Tahmini

- Bakım tahmini, sistemin hangi bölümlerinin sorun çıkarabileceğinin ve yüksek bakım maliyetine sahip olacağının değerlendirmesidir
- Değişimin kabulü, değişimden etkilenen bileşenlerin bakım kolaylığına bağlıdır
- Değişiklikleri gerçekleştirmek sistemin yapısını bozar ve bakım kolaylığını azaltır
- Bakım maliyeti değişiklik sayısına ve değişikliklerin zorluğuna bağlıdır

Bakım Tahmini



Değişim Tahmini

- Değişiklik sayısını tahmin etmek sistem içindeki ve sistemin çevresi ile arasındaki ilişkileri anlamayı gerektirir
- Sıkıca bağlı sistemlerin ortam her değiştiğinde değiştirilmeleri gerekir
- Bu ilişkileri etkileyen faktörler
  - Sistem arayüzlerinin sayısı ve karmaşıklığı
  - Doğal olarak değişen sistem gereksinimlerinin sayısı
  - Sistemin kullanıldığı iş süreçleri

Karmaşıklık Ölçütleri

- Bakım tahminleri sistem bileşenlerinin karmaşıklığı değerlendirilerek yapılabilir
- Çalışmalar, en fazla bakım emeğinin sistemin az bir bileşeni üzerinde harcandığını göstermektedir
- Karmaşıklık şunlara bağlıdır
  - Kontrol yapılarının karmaşıklığı
  - Veri yapılarının karmaşıklığı
  - Nesne, metot ve modül boyutu

Süreç Ölçütleri

- Süreç ölçütleri bakım kolaylığını değerlendirmek için kullanılabilir
  - Düzeltici bakım için gelen istek sayısı
  - Etki analizi için gereken ortalama süre
  - Değişikliği gerçekleştirmek için gereken ortalama süre
  - Öncelikli değişim isteklerinin sayısı
- Eğer bunlardan bazıları veya hepsi artıyorsa bakım kolaylığının azaldığı söylenebilir

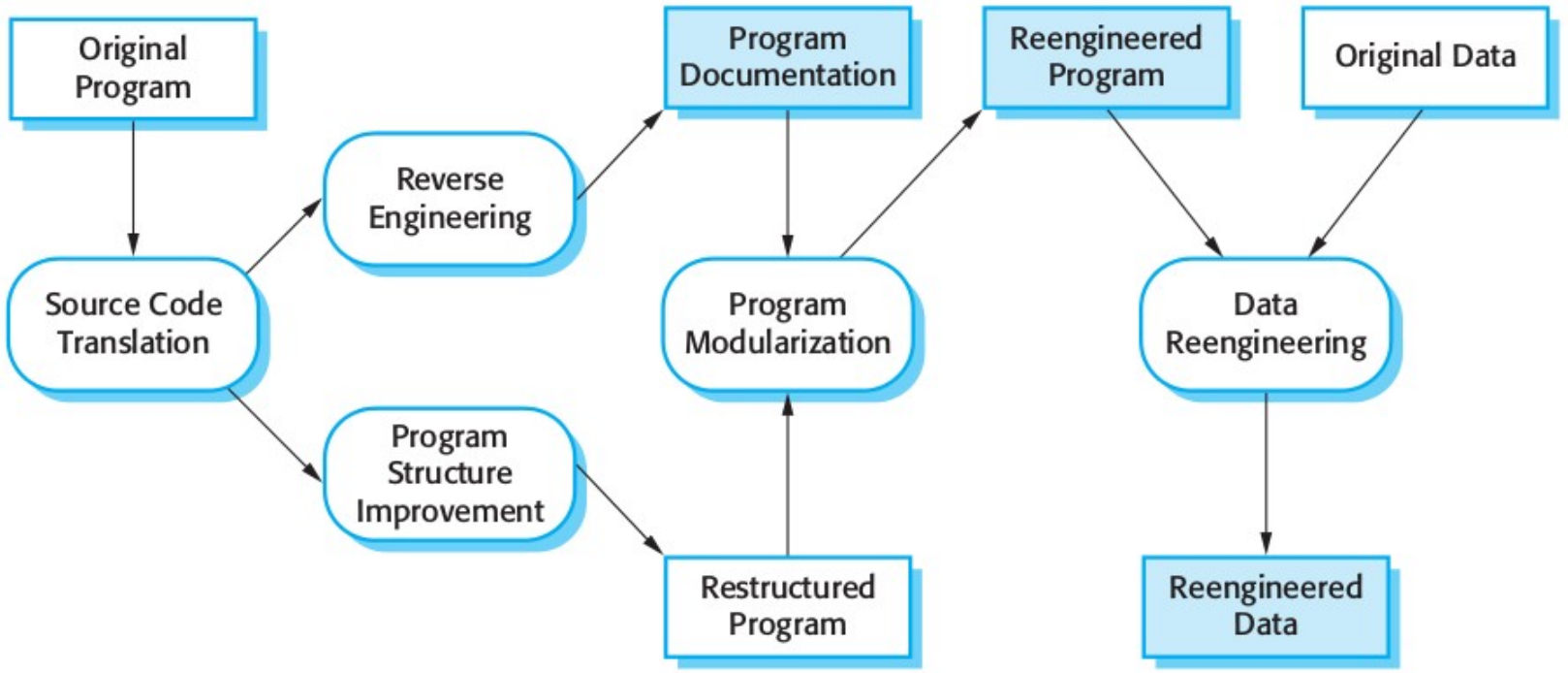
Sistemin Yeniden Yapılandırılması

- Eski sistemin işlevselliğini değiştirmeden bir kısmının yeniden yapılandırılması veya yeniden yazılmasıdır
- Büyük bir sisteme ait bazı alt sistemlerin sık sık bakıma ihtiyaç duyması halinde yapılabilir
- Yeniden yapılandırma bakım kolaylığını arttırmaya yönelik emek içerir. Sistem yeniden yapılandırılabilir veya yeniden dokümanlar oluşturulabilir

Yeniden Yapılandırmanın Avantajları

- Azaltılmış risk
  - Yeni yazılım geliştirme işi yüksek risk içerir. Geliştirme problemleri, çalışanlarla ilgili problemler olabilir
- Azaltılmış maliyet
  - Yeniden yapılandırmanın maliyeti yeni yazılım geliştirme maliyetinden oldukça düşüktür

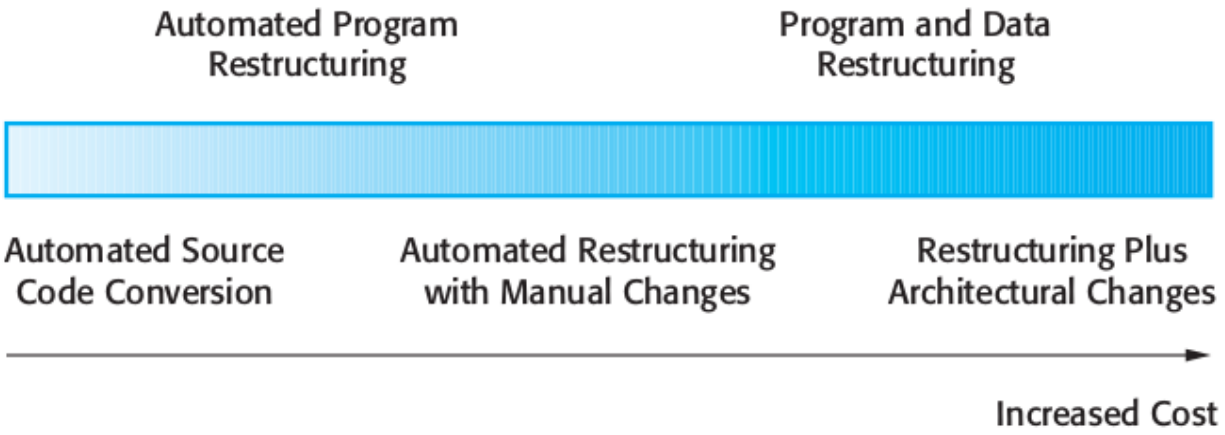
Yeniden Yapılandırma Süreci



Yeniden Yapılandırma Süreci İşleri

- Kaynak kod çevrimi
  - Kod yeni bir dile çevrilir
- Geri mühendislik
  - Programı anlamak için analiz etme
- Program yapısını iyileştirme
  - Daha iyi anlaşılması için yeniden yapılandırma
- Program modülerleştirme
  - Program yapısını yeniden düzenleme
- Veriyi yeniden yapılandırma
  - Sistem verisini temizleme ve yeniden yapılandırma

Yeniden Yapılandırma Yaklaşımları



Refactoring ile Önleyici Bakım

- Refactoring, değişiklikler yapıldıkça sistem yapısının bozulmasını yavaşlatmak için programda iyileştirmeler yapılmasıdır
- Gelecekte yapılacak değişikliklerin yaratacağı problemleri azalttığı için refactoring'i önleyici bakım olrak nitelendirebiliriz
- Refactoring, programın yapısını iyileştirmek için değişiklik yapmayı ve anlaşılmasını kolaylaştırmak için karmaşıklığını azaltmayı içerir
- Bir programı refactor ettiğinizde yeni işlevsellik eklememeli sadece programın iyileştirilmesine odaklanmalısınız

Yeniden Yapılandırma ve Refactoring

- Yeniden yapılandırma, sistem bir süredir bakım altında olduğunda ve bakım maliyetleri giderek arttığında yapılır. Bakımı daha kolay olan yeni bir siste yaratmak için, eski sistemi işleyecek ve yeniden yapılandıracak otomatik araçlar kullanırsınız
- Refactoring, sistem geliştirme ve evrim sürecinde yapılan sürekli iyileştirmelerdir. Sistemin bakım maliyetini artıracak ve bakımını zorlaştıracak kodda ve program yapısında oluşabilecek bozulmalardan kaçınmak amacıyla yapılır

Koddan Gelen “kötü kokular”

- Tekrarlı kod
  - Aynı veya çok benzer kod programın farklı parçalarında bulunabilir. Bu kod kaldırılabilir ve onun yerine aynı işi yapan bir metot yazılabilir ve gerektiği yerde metot çağrılır
- Uzun metotlar
  - Eğer bir metot çok uzunsa, daha kısa metotlara bölünebilir
- Switch (case) yapıları
  - Burada genellikle kod tekrarı vardır. Switch yapıları programın farklı yerlerine dağılmış olabilir. Nesneye dayalı dillerde çok biçimlilik kullanarak aynı işlevi gerçekleştirebilirsiniz

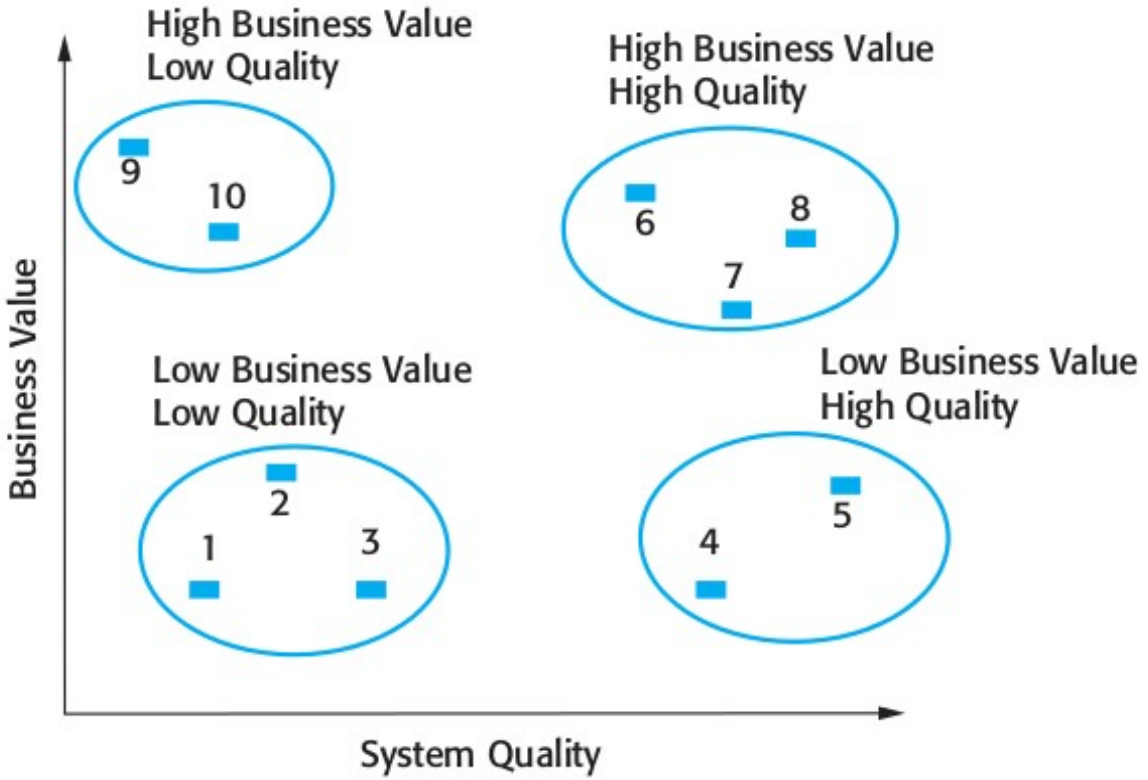
Koddan Gelen “kötü kokular”

- Veri kümelenmesi
  - Aynı veri grubu programda birçok yerde görünüyorsa (sınıfların field'ları metotların parametreleri). Bunlar tüm veriyi kapsülleyecek bir nesne ile değiştirilebilirler
- Kurgusal genellik
  - Yazılım geliştiriciler gelecekte ihtiyaç duyulabileceğini düşünerek programa bazı genellemeler eklerler. Bu kaldırılabilir

Eski Sistem Yönetimi

- Eski sisteme dayalı olarak çalışan kuruluşlar bu sistemlerin evrimi için bir strateji belirlemelidir
  - Sistemi hurdaya çıkarmak
  - Sistemin bakımına devam etmek
  - Yeniden yapılandırma ile sistemin bakım kolaylığını arttırmak
  - Sistemi yeni bir sistemle değiştirmek
- Seçilen strateji sistem kalitesine ve iş değerine bağlı olmalıdır

Eski Sistemi Değerlendirme Örneği



Eski Sistem Kategorileri

- Düşük kalite, düşük iş değeri
  - Sistemi hurdaya çıkar
- Düşük kalite, yüksek iş değeri
  - Sistemi yeniden yapılandır veya mümkünse uygun bir COTS ile değiştir
- Yüksek kalite, düşük iş değeri
  - Kullanmaya devam et, bakım çok masraflıysa hurdaya çıkar
- Yüksek kalite, yüksek iş değeri
  - Normal sistem bakımı ile kullanmaya devam et

İş Değeri Belirleme

- Değerlendirme farklı bakış açılarını dikkate almalıdır
  - Sistemin son kullanıcıları
  - İş müşterileri
  - Bölüm yöneticileri
  - IT yöneticileri
  - Kıdemli yöneticiler
- Farklı paydaşlarla görüşmeler yapıp sonuçları birleştirmek gerek

İş Değeri Belirleme Konuları

- Sistemin kullanımı
  - Eğer sistem seyrek olarak veya çok az sayıda kişi tarafından kullanılıyorsa düşük iş değerine sahip olabilirler
- Desteklenen iş süreçleri
  - Bir sistem etkin olmayan iş süreçlerini kullanmaya zorluyorsa iş değeri düşük olabilir
- Sistem güvenilirliği
  - Eğer sistem güvenilir değilse ve problemler müşteriyi doğrudan etkiliyorsa sistem düşük iş değerine sahiptir
- Sistem çıktıları
  - Eğer iş sistem çıktılarına bağlıysa sistemin iş değeri yüksektir

Sistem Kalite Değerlendirmesi

- İş süreci değerlendirme
- İş süreçleri hedefleri ne kadar destekliyor?
- Çevresel değerlendirme
  - Sistemin çevresi ne kadar etkin ve bakımı ne kadar masraflı?
- Uygulama değerlendirme
  - Yazılım sisteminin kalitesi nedir?

İş Süreci Değerlendirme

- Bakış açısı odaklı yaklaşım kullanın ve sistem paydaşlarından cevaplar almaya çalışın
  - Tanımlanmış bir süreç modeli var mı ve bu model takip ediliyor mu?
  - Kurumun farklı bölümlerinde aynı iş için farklı süreçler mi kullanılıyor?
  - Süreç nasıl uyarlandı?
  - Diğer iş süreçleriyle ilişkiler nasıl ve bunlar gerekli mi?
  - Süreç, kullanılan sistem tarafından etkin bir şekilde destekleniyor mu?
- Örnek – web-tabanlı sipariş çok yaygın kullanıldığından bir seyahat satın alma uygulaması düşük iş değerine sahip olabilir

Çevresel Değerlendirme Faktörleri

Faktör	Sorular
Üretici istikrarı	Üretici piyasada hala var mı? Üretici mali olarak istikrarlı durumda mı ve varlığını sürdürebilecek gibi görünüyor mu? Eğer üretici artık yoksa, sistemlerin bakımını yapan başka birileri var mı?
Hata oranı	Donanımın rapor edilen hata oranı yüksek mi? Yazılım göçüyor ve sistemin yeniden başlatılması gerekiyor mu?
Yaş	Donanım ve yazılım kaç yaşında? Yazılım ve donanım ne kadar eskiyse o kadar kullanılmaz durumda olabilir. Hala çalışıyor olabilirler ama modern bir sisteme geçmek ekonomik ve iş açısından çok daha faydalı olabilir
Performans	Sistemin performansı yeterli mi? Performans sistem kullanıcıları üzerinde önemli etkilere sahip mi?

Çevresel Değerlendirme Faktörleri

Faktör	Sorular
Destek gereksinimleri	Donanım ve yazılım ihtiyacı olan yerel destekler nelerdir? Eğer bu destekler yüksek maliyetliyse sistemi yeni bir sistemle değiştirme düşünülebilir
Bakım giderleri	Donanım bakımı ve yazılım lisanslarının maliyeti nedir? Eski donanım modern sistemlerden daha fazla bakım maliyetine sahip olabilir. Yazılımın yıllık lisas ücretleri de yüksek olabilir
Uyumlu çalışma	Sistemlerin başka sistemlerle çalışmasında problemler var mı? Örneğin derleyiciler, işletim sisteminin mevcut sürümü ile uyumlu çalışabiliyor mu? Donanım simülatörü gerekli mi?

Çevresel Değerlendirme Faktörleri

Faktör	Sorular
Anlaşılabilirlik	Mevcut sistemin kodunu anlamak ne kadar zor?Kullanılan kontrol yapıları ne kadar karmaşık? Değişken isimleri yaptıkları işi yansıtan anlamlı isimler mi?
Belgeleme	Hangi sistem belgeleri var? Belgeler tam, uyumlu ve güncel mi?
Veri	Sistem için açık bir veri modeli var mı? Veri hangi amaçla tekrarlı olarak tutuluyor? Veri uyumlu ve güncel mi?

Çevresel Değerlendirme Faktörleri

Faktör	Sorular
Programlama Dili	Sistemi geliştirmek için kullanılan dile ait modern derleyiciler var mı? Programlama dili hala yeni sistemler geliştirmek için kullanılıyor mu?
Konfigürasyon Yönetimi	Sistemin tüm parçaları ve versiyonları bir yönetim aracı tarafından takip ediliyor mu? Mevcut sistemde kullanılan bileşenlerin versiyonlarının açık tanımı var mı?
Test Verisi	Sistem için test verisi var mı? Sisteme yeni özellikler eklenirken yapılan regresyon testlerinin kayıtları var mı?
Personel Yetenekleri	Uygulamanın bakımını yapabilecek yetenekte insanlar var mı? Sistemde deneyimi olan insanlar var mı?

Sistem Ölçümü

- Sistemin kalitesini değerlendirmek için nicel veri kullanılabilir
  - Değişiklik taleplerinin sayısı
  - Sistem tarafından kullanılan kullanıcı arayüzlerinin sayısı
  - Sistem tarafından kullanılan veri hacmi

Önemli Noktalar

- Tüç tipte sistem bakımı vardır; hata düzeltme, sistemi yeni ortamda çalışması için değiştirmeve yeni veya değişen gereksinimleri gerçekleştirme
- Yazılımın yeniden yapılandırılması yazılımın daha kolay anlaşılması ve değiştrilmesi için yapısal değişiklikler yapma ve yeniden dokümanlama işleri ile ilgilidir
- Refactoring, programın işlevselliğini koruyarak programda değişiklikler yapmaktır ve bir tür önleyici bakımdır
- Kullanılmakta olan sistemlerin iş değeri ve kalitesi, sistemin başka sistemle değiştirilmesine, dönüşümüne veya kullanılmaya devam etmesine karar vermek için değerlendirilmesi gereken unsurlardır