**BLM 426 YAZILIM MÜHENDİSLİĞİ**

**BAHAR 2016**

**Yrd. Doç. Dr. Nesrin AYDIN ATASOY**

**12. HAFTA: YAZILIM BAKIMI**

Bilgisayar tabanlı sistemlerin tasarlanıp geliştirilmesinden ve kullanıcıya tesliminden sonra bakım (maintenance) aşaması başlar. Sistemin donanım öğelerinin bakımı, temizleme, eskimiş ya da arızalanmış parçaların değiştirilmesi şeklinde yapılır veya tüm donanım öğesi yenisiyle değiştirilir. Yazılım öğelerinde ise bütün eskime ya da arızalanma durumu yoktur. Zaman içinde ortaya çıkan yeni isterlerin karşılanması veya bulunan hataların giderilmesi çalışmaları yazılım bakımının temelini oluşturur. Yazılım bakımı, yazılım geliştirilmesinde daha fazla zaman ve iş gücü gerektirebilir.

Yazılım bakımı bir aracın bakımı gibi periyodik olmayıp gelişen koşullara göre ortaya çıkar. Örneğin, daha önce hazırlanmış bir yazılımın değişen teknoloji nedeniyle yeni bilgisayar mimarileri ve işletim sistemlerine uyarlanması önemli bir bakım işidir.

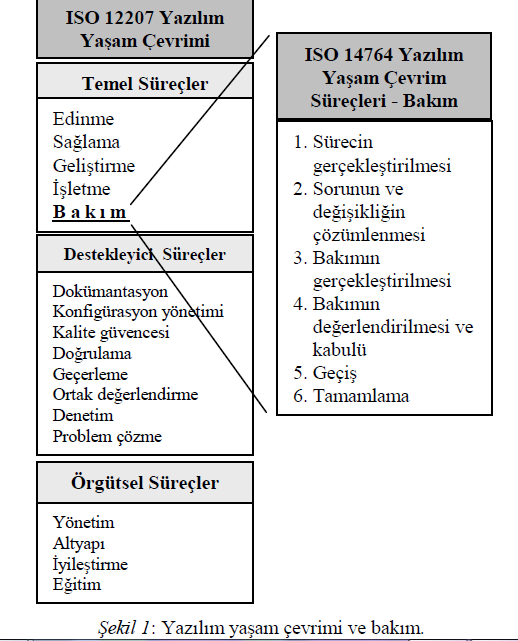
Değişiklik, bilgisayar tabanlı sistemler için kaçınılmaz bir gerçektir. O halde, değişiklikleri değerlendirmek, denetim altında tutmak ve uygulamak için bir düzen geliştirilmeli ve ona uyulmalıdır. Yazılım bakımı ile ilgili planlar ve içerikleri de IEEE/ EIA 12207 standardı içinde yer almaktadır. Bu bölümde yazılım bakımının nasıl yapılacağı hakkında temel bilgiler vereceğiz.

# Yazılım Bakımının Temelleri

Yazılım bakımı, müşteri ile geliştirici arasında yapılan sözleşmeye göre bazen yazılım geliştirme sürecinde yer almamaktadır. Oysa günümüzde pek çok uygulama alanı değişen isterlere sahiptir. Bu nedenle de yazılım geliştirilip kullanıma sunulduktan sonra dahi büyük miktarda değişiklik ve hata düzeltme istekleri oluşabilmektedir.

## Bakım Türleri

Bakım işleri yalnızca hataları düzeltmek olmayıp ürünün tesliminden sonra yapılması gereken çeşitli türde işleri de kapsar. Bakımın türleri olarak görebileceğimiz bu işlerin neler olduğuna bir bakalım:



**Şekil 1.** Yazılım yaşam çevrimi ve bakım.

## Düzeltici Bakım:

Yazılım testi her zaman tüm kusurların bulunmasını ve giderilmesini sağlamaz. Çalışan bir yazılımda her an hata bulunması olasılığı vardır. Bir kısım yazılım kusuru ancak kullanım sırasında ortaya çıkar. Bu kusurların giderilmesi için geliştiriciye bilgi verilir. Kusurun sebebini araştırmaya ve gidermeye yönelik işlere düzeltici (corrective) bakım adı verilir. Kullanım sırasında bulunan ve raporlanan yazılım kusurlarının giderilmesi, hatanın önemine bağlı olarak, ya hemen yapılır yada birkaç tanesiyle beraberce ele alınıp düzeltmeler uygulanır ve yeni bir sürüm çıkarılır. Bu bakım geliştiricinin bakım aşamasında mutlaka yürütmesi gereken bir etkinliktir.

## Uyarlayıcı Bakım:

Bilgi işleme dünyasındaki hızlı değişmeler ve teknolojik gelişmeler nedeniyle yazılımın yeni donanıma, işletim sistemlerine, bunların yeni sürümlerine, yeni uç birimlere göre uyarlanması, sürümünün yükseltilmesi ve güncelleştirilmesi işlevine uyarlayıcı (adaptive) bakım denmektedir.

Yazılımı kullanılan işletmeler zaman içinde çeşitli nedenlerle işlerin yürütülme şeklinde, kullanılan yöntemlerde (örneğin yazıcıdan alınan kağıt baskının biçimince değişiklik gerekmesi) değişiklikler yapma gereksinimi duyabilirler.

Son yıllardaki gelişmelere bakarak, bir yazılımın ortalama ömrünün 10 yıl olduğunu, fakat donanım birimlerinin yürürlükte kalma sürelerinin 1-2 yıl olduğunu görmekteyiz. Bu durumda, ya yazılımlar eski teknoloji ile kullanılmaya devam edilecek ya da taşınabilirlik özelliği ön planda tutularak güncel teknoloji kullanılacaktır. Güncel olmayan teknolojiyi kullanmanın temel sorunu ise ek donanım gereksinimlerini, yedek parçaların sağlanmasında veya yeni ortaya çıkan kullanışlı yazılım araçlarının sisteme eklenmesinde güçlüklerle karşılaşmasıdır.

## İyileştirici Bakım:

Yazılım geliştirilerek test edilerek başarılı bir şekilde kullanıcıya sunulduktan sonra, yeni işlevler eklemek, var olanlara başarımı ve verimi artırıcı düzenlemeler yapmak iyileştirici (perfective) bakım işleri içinde yer alır. Bu şekilde yazılımın yeni sürümleri ortaya çıkar ve kullanıcının hizmetine sunulur. Örneğin veri tabanı erişimi ortalama 5 saniye olan bir yazılımda yeni bir arama algoritmasının kullanımı ile arama süresi 3 saniyeye indirebiliyorsa bu değişikliğin uygulanması bir iyileştirici bakımdır. Yine, kullanımdaki bir yazılıma, yeni kullanıcı isterlerine göre yeni işlevler kazandırmak iyileştirici bakım kapsamına girer. Bu amaçla, ticari yazılım dünyasında paket yazılımların yeni özellikler içeren yeni sürümleri sürekli olarak piyasaya sürülmektedir. Özel geliştirilmiş sistemlerin yazılım bakım sözleşmelerinde de bu konu dikkate alınarak genellikle iyileştirici bakım için ayrı sorumluluklar tanımlanır, maliyetler hesaplanır.

## Önleyici Bakım:

Yazılımın gelecekte uygulanabilecek değişikliklere daha iyi bir temel oluşturması bakılabilirliğinin ve güvenirliliğinin arttırılması için ön tedbir niteliğindeki işlemler önleyici (preventive) bakım kapsamına girmektedir. Örneğin sık sık değişiklik yapılması gereken bir modülün tasarımını daha esnek bir hale getirmek üzere yeni bir geliştirme süreci uygulanması, bunun sonunda oluşan yeni sürümün müşteriye verilmesi ve sonraki değişikliklerin daha kolay yapılmasını sağlayan önleyici bir bakımdır. Bu tür bakım geliştiricinin uzun süreli bakım anlaşması kapsamında yer alabilir.

## Bakım Ekibi

Genel olarak yazılım geliştiriciler bakım işleri için özel bir ekip ayırmazlar. Ancak, yazılımla ilgili sorun veya yeni gereksinimlerin değerlendirilmesi ve önceliklendirilmesinde belirli bir ekibin tanımlanması gereklidir.

Bakım ekibinin yapısı önemlidir. Bu yapıda daha önce geliştirilmiş sistem hakkında bilgisi olan en az bir teknik danışman bulunur. Bu kişi sistemin bileşenlerini, teknik özelliklerini bilen birisidir. Kullanıcın bakım istekleri bir bakım denetçisine iletilir. Bakım denetçisi, o proje ile ilgilenmiş olan teknik danışmanından faydalanarak bir değişiklik önerisi oluşturur. Bu öneri, Değişiklik Kontrol Kurulu (Change Control Board) ‘nda görüşülür. Öneri kabul edilmezse kullanıcı durumdan haberdar edilir. Önerinin kabul edilmesi durumunda kurul tarafından o değişiklik için bir sorumlu personel atanır. Sorumlu personel bir iş planı yaparak bakım personelini görevlendirir. Düzenleşim yönetimi ile eşgüdüm halinde bakım personelinin yeni bir sürüm oluşturulması sağlanır. Yeni sürüm gerekli testlerden geçirildikten sonra teslim personeli tarafından eğitimle birlikte kullanıcıya verilir.

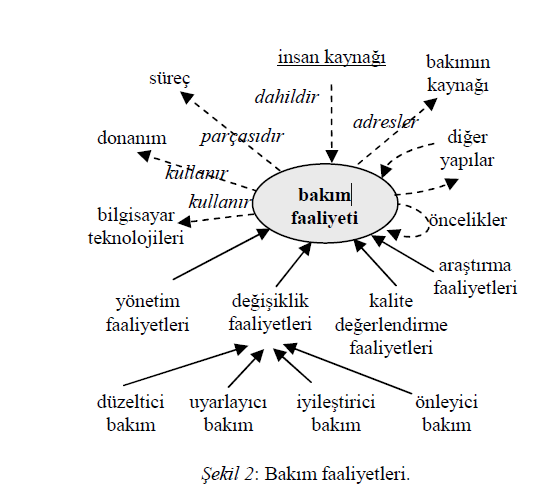
Kullanıcın bildirdiği yazılımla ilgili sorun ve yeni isterleri bakım denetleyicisi yerine geliştiricinin teknik destek birimi, atanmış ürün yöneticisi veya o projenin yöneticisi toplayabilir.

Küçük ölçekli bir geliştirme grubunda Bakım Denetçisi ile Değişiklik Kontrol Kurulu ayni kişi olabilir. Bu görevi proje yöneticisi veya ürün yöneticisi de yapabilir. Büyük projelerde ise bu kurul yöneticilerden ve kıdemli teknik personelden oluşur.

Sorumluluklar projenin bakım evresinden önce dağıtılır ve uygun ekip yapısı sağlanırsa, gerçekten bir bakım isteği ortaya çıktığında oluşabilecek karmaşa önlenmiş olur. Daha da önemlisi, sorumlulukların önceden tanımlanmasıyla personel başka bir geliştirme işinde görevliyken, onu oradan alıp acil bir bakım işi için görevlendirmek gerektiğinde oluşabilecek olumsuz durumların önüne geçilir.

# Bakım Aşamaları

Bakım evresinde bulunan yazılım için bir bakım işi ortaya çıktığında geliştirici tarafından standart bir süreç izlenmelidir. Bu sürecin aşamaları tıpkı bir yazılım geliştirme sürecinde benzer. Tek farkı, hazır olan bir belge ve kod yapısı üzerinde değişikliklerin uygulanmasıdır. Bakım aşamalarını şu şekilde özetleyebiliriz.



**Şekil 2.** Bakım faaliyetleri.

**İsterler Çözümlenmesi:** Bu aşamada sorun veya değişiklik tanımlanır ve sınıflandırılır. Sistemde beklenen yeni düzenlemelere ve işlevlere ilişkin isterler tanımlanır. Buna göre var olan Yazılım İsterleri Belirtimi ve buna paralel olarak da Yazılım Test Tanımlaması belgesi güncellenir.

**Tasarım:** Eldeki tasarım gözden geçirilerek yeni istekler eklenir ve Yazılım Tasarım Tanımlaması belgesi güncellenir.

**Gerçekleştirim:** Yeni tasarım koda yansıtılarak gerekli kod değiştirme veya yeniden modül geliştirme yapılır. Birim testleri gerekiyorsa yeni eklenen testlerle yapılarak tümleştirmeye hazır hale getirilir. Yazılım öğeleri kendi aralarında sonra da donanım ile tümleştirilir.

**Test:** Yeni eklenen testlerin yanında, sistematik olarak seçilen belirli testler tekrar edilerek tüm yazılımın testi gerçekleştirilir. Bu amaçla yapılan testlere regresyon testi denir. Sistemin genelinin güvenilirliğini kanıtlamak üzere müşteri önünde kabul testleri yapılır.

**Teslim:** Yazılımın yeni bir sürüm olarak müşteriye verilir. Büyük ölçekli yazılımlarda, yalnızca değişikliğe uğramış kısımlar için kurulum ve çalıştırma kılavuzu ile beraber yama (patch) üretilir. Teslim edilir ve kullanım kılavuzları güncellenir.

## Raporlama

Genel olarak, yazılımda bir değişiklik yapılması için istekler Değişiklik Önerisi (Change Proposal) ile, sorunlar da Yazılım Sorun Raporu (Software Trouble Report) ile bildirilir. Değişiklik Önerisi, kullanıcı tarafından yazılımda ne tür değişiklik arzu ettiğini açıklayacak şekilde hazırlanabileceği gibi geliştirici personel tarafından bir öneri şeklinde de hazırlanabilir.

Kullanıcının yeni isteğini karşılamak üzere ya da kendileri tarafından belirlenmiş bir iyileştirme amacıyla verilen Değişiklik Önerisi en az aşağıdaki bilgileri içermelidir:

* Sistem veya alt sistem adı, öğe adı
* Değişikliğin tanımı
* Değişikliğin yapılacağı öğe, bileşen veya birim
* Yapılan değişiklikten etkilenmesi olası diğer öğe, bileşen veya birimler
* Değişiklikleri yapabilmek için harcanması gereken tahmini işgücü
* İsteğin öncelik derecesi
* İsteğin numarası (takip edebilmek için)
* Diğer açıklayıcı bilgiler
* Öneriyi inceleyen makamların imzaları ve tarihleri
* Karar (inceleme sonunda oluşturulur)

Herhangi bir yazılım kusurunu raporlamak için kullanılan Yazılım Sorun Raporu da benzer şekilde olup en az aşağıdaki bilgileri içermelidir:

* Sistem veya alt sistem adı, öğe adı
* Sorunun açık tanımı (mutlaka kesin ifadeler kullanılmalıdır)
* Hatalı duruma düşülmesine neden olan giriş verileri
* Varsa raporlama sisteminin oluşturduğu kayıtlar (log data)
* Sorunu rapor eden kişiye ait erişim bilgileri
* Sorun Raporu numarası (takip edebilmek için)
* Sorunu daha ayrıntılı inceleyen mühendislere ait erişim bilgileri
* Raporu inceleyen makamların imzaları ve tarihleri
* Karar (inceleme sonunda oluşturulur)

Bazen bu iki belge birleştirilir ve boşlukların doldurulacağı bir tablo şeklinde tek bir şablon kullanılır. Bu belgeler bir bilgisayarda veya elle doldurulur.

# Bakım Kolaylığı

Nitelikli bir yazılımın önemli özelliklerinden biri bakım kolaylığıdır. Bu özelliği, yazılımın anlaşılmasındaki, düzeltilmesindeki, iyileştirme ve uyarlanmasındaki kolaylık olarak tanımlayabiliriz. Teknik olarak, her türlü değişiklik istemini gerçekleştirmek mümkündür. Ancak, önemli olan bunu en düşük maliyetle, en kısa zamanda, doğru olarak ve yazılımın niteliklerini bozmadan yapabilmektir. Unutulmamalıdır ki, bir kez değişiklik isteminde bulunan yazılım, gelecekte de değişikliğe uğrayabilir. Bu nedenle, bakıma etki eden çeşitli etmenler dikkate alınmalı, bakım işleri belirli bir nitelik güvence çerçevesi içinde yürütülmeli ve gelecekte de yararlı olabilecek niceliksel değerler toplanmalıdır.

## Denetim Etmenleri

Yazılım bakımının önemi, hem geliştirici hem de müşteri tarafından uzun vadede daha iyi anlaşılır. Bu önemin iyi kavranması da yazılım bakımını denetleyen çeşitli etmenlere bağlıdır. Bunları şu şekilde gruplamak mümkündür.

## Geliştirme ortamına bağlı olanlar

* + Yazılım geliştirme personelinin varlığı
  + Sistem yapısının anlaşılabilirliği
  + Sistemin işletim kolaylığı
  + Standart bilgisayar donanımı kullanımı (ana, test, hedef sistem)
  + Standart işletim sistemi kullanımı
  + Standart programlama dillerinin kullanımı
  + Standart veya taşınabilir kütüphanelerin kullanımı
  + Standart belgelendirme yöntemlerinin kullanımı
  + Test durumlarının yeniden oluşturabilme olanağı
  + Hata ayıklama gereçlerinin varlığı

## Personele bağlı olanlar

* + Personelin aynı konudaki geliştirme deneyimi
  + Kullanılan geliştirme sürecindeki deneyim
  + Personelin iş disiplini
  + Personelin uygulama alanı hakkındaki bilgisi
  + Bakım evresine giren yazılım hakkında deneyimi olan personelin bulunması

## Müşteriye bağlı olanlar

* + Kullanıcıların tek veya çok sayıda olması (tek müşterinin siparişi veya kitle kullanımı amaçlı olması)
  + Müşteri ile geliştiricinin yazılımın yaşam çevrimi boyunca olan ilişkisi (garanti süresi, teknik destek anlaşmaları, yeni değişiklik isteklerinin çıkma olasılığı gibi)
  + Müşteri isteklerinin değişme sıklığı
  + Müşterinin ortam değiştirme sıklığı
  + Uygulama alanı isterlerinin değişme sıklığı

## 

**Şekil 3.** Bakım faaliyetlerine ait insan varlıkları.

# Bakımın Niteliği

Bir yazılım projesinin nitelikli bir bakım evresine sahip olabilmesi için işin başında aşağıdaki noktalara önem verilmesi gereklidir:

* Bakım istekleri her zaman resmi bir Değişim Denetim Süreci kapsamında ele alınmalıdır.
* Bakım istekleri yeterince hızlı olarak yerine getirilebilmelidir.
* Bakım istekleri ve çalışmaları sistematik olarak izlenebilmelidir.
* Bakım işlemleri her zaman resmi olarak tanımlanmalı ve yönetilmeli, gelişigüzel bir şekilde ele alınmamalıdır.
* Bakım sırasında belgelendirme standartlarına uyulmalıdır.
* Uygun nitelikte bakım personeli kullanılmalıdır.

## Niceliksel Ölçümler

Yazılımın bakım kolaylığı, nitelik ve güvenilirlik gibi ölçülmesi oldukça zor olan bir özelliktir. Ancak, bakım işlemlerinin özelliklerini dikkate alarak bazı metrikler ve niceliksel ölçümler yapmak mümkündür. Bu metriklerin önemli ve yaygın olanları şunlardır:

* Sorunun belirlenip rapor edilme süresi
* İdari işlemlerdeki gecikme süresi
* Değişikliğin tanımlanması için gereken süre
* Sorun bildirimlerinin müşterilere göre dağılımı
* Sorun bildirimlerinin hata türlerine göre dağılımı
* Sorun bildirimlerinin modüllere göre dağılımı
* Düzeltme için geçen zaman
* Yerel test süresi
* Kullanım yerindeki test süresi
* Toplam süre
* Toplam işgücü ve maliyet

Bu metriklerin hemen hemen hepsi bakım işleri sırasında zorluk çekmeden kaydedilebilir. Kaydedilen veriler yöneticilere kullandıkları yöntemlerin ve araçların etkinliği, olası maliyet kestirimi konusunda bilgi verilir.

Yazılım bakım işlerinin değerlendirilebilmesi için bahsettiğimiz bu metriklerin toplanmış ve kayıt altına alınmış olması gereklidir. Özellikle, uzun süre kullanılması düşünülen yazılımların bakım evrelerinin sağlıklı olarak devam edebilmesi için her bakım işinde bu bilgiler toplanmalı ve program veya proje tanımlayıcısına (isim veya kod numarası) göre saklanmalıdır. Görev alan personel bilgileri ile bakım işi sonunda elde edilen deneyimler de kaydedilmesi gerekenler arasında yer almalıdır.

# Bakım Sorunları

Bakım sırasında karşılaşılan sorunları bilmek ona karşı hazırlıklı olabilmek için yarar sağlayacaktır. Genellikle, geliştirilen yazılımın bakılabilirlik özelliğinin eksikliği, bir disiplin içinde geliştirilmiş olmaması sorunların kaynağını oluşturur. Bu sorunları şu şekilde listeleyebiliriz:

* Yazılımın çok fazla sürümü ortaya çıkarsa, bakımı da o derece güçleşir. Hatta gerekli değişikliklerin tam olarak kaydedilememiş olması durumunda bakımı yapabilmek mümkün olmaz.
* Yazılımın geliştirildiği süreci aynen takip etmek zaman ve işgücü açısından çoğu zaman olanaksızdır.
* Bir başkasının yazdığı kodu anlamak genellikle çok zaman alır. Eğer belgelendirme ve kod içindeki açıklamalar yetersiz ise ciddi sorunlar ortaya çıkar.
* Belgelendirme yetersiz, eksik, hatalı olabilir ya da hiç olmayabilir. Bu durumda yalnızca kodu okuyup anlamak gerekecektir.
* Personel devamlılığı her zaman karşılaşılması muhtemel genel bir sorundur.

## Geliştirici İçin Kurallar

Yüksek nitelikli yazılım üretmeyi amaçlayan geliştiricilerin bakım evresinde uygulaması gerekli kurallar arasında şunları sayabiliriz:

* Bakım isteklerinin durumu sistematik olarak izlenebilmelidir.
* Bakım istekleri her zaman resmi bir Değişiklik Denetim Süreci kapsamında ele alınmalıdır.
* Bakım istekleri için süre ve maliyet kestirimi doğru olarak yapılabilmelidir.
* Müşterinin yazılımla ilgili bildirdiği sorun ve yeni gereksinimler teknik destek birimi, düzenleşim yöneticisi, ürün yöneticisi veya proje yöneticisinin görevlendireceği bir kişi ya da ekip tarafından toplanmalıdır.

# Bakımın Yan Etkileri

Yazılımın bakımı genellikle biraz “tehlike” içerir. Zira, test edilmiş ve kullanıma sunulmuş bir üründe yapılacak küçük sayılabilen bir değişiklik iyi çözümlenmez ve etkileri iyi takip edilmezse daha büyük yan etkilere yol açabilir. Karmaşık bir algoritma içinde yer alan bir tek değişkenin değerinin değiştirilmesi ya da yeni bir deyim eklenmesi gibi küçük sayılabilecek bir değişiklik, tüm yazılımın hatalı çalışmasına veya çökmesine yol açabilir. Tasarımın ve kodun iyi belgelendirilmesi, dikkatli bir bağlanım testi bu tür sonuçları azaltabilir. Genellikle, kod içinde, verilerde, belgelemede ve başarımda ortaya çıkan bu istenmeyen sonuçlara bakımın yan etkileri denmektedir.

* Kodlamanın Etkilenmesi
* Verilerin Etkilenmesi
* Belgelendirmenin Etkilendirmesi
* Başarımın Etkilenmesi

# Belgelendirilmemiş Yazılımların Bakımı

Yazılım geliştirmekle görevli grupların baş etmeleri zor olan bir sorun da hiçbir kaydı ve belgesi bulunmayan eski yazılımların bakımını yürütmektir. Çeşitli nedenlerden dolayı on ya da yirmi yıl önce yazılmış kodları bugün tekrar kullanmak, yeni ortamlara uyarlamak veya değişiklik yaparak kullanmaya devam etmek gerekebilir. Ancak, bu yazılımları geliştirmiş olanlar artık o grupta, hatta o firma ya da kurumda bulunmayabilir. Geliştirme sırasında belirli bir yöntembilim uygulanmamış, yapılması gerekli olan belgelendirme de çok az yapılmış ya da hiç yapılmamış olabilir. Böyle durumlarda bakım için uygulanacak yöntemler şunlar olabilir:

* Aynı kod üzerinde değişiklik yapılabilir.
* Kod başka ortamlara taşınabilir.
* Tersine mühendislik yapılabilir.
* Kaynak kodu kapsayıcı içine almak
* Yeni bir dille yeniden yazmak

## Tersine Mühendislik

Tersine mühendislik (reverse enginerring) temel olarak elektronik ya da mekanik mühendisliğine uygulanmaya başlanmıştır. Üreticisi dışında başa geliştiriciler, ticari casusluk içi bir elektronik sistem donanımının veya mekanik parçaların nasıl çalıştığını, ürünün nasıl üretildiğini anlamak üzere inceleme ve testler yaparlar. Üretim sırlarını ortaya çıkarmaya çalışırlar. Bu şekilde bir ürünün kopyasını elde etmeye veya yenisini geliştirmeye gayret ederler.

Yazılımda uygulanan tersine mühendislik de aynı sayılır. Üzerinde çalışılan ürün başka bir geliştiriciye ait olabileceği gibi aynı geliştiriciye ait daha önceden üretilmiş yazılımlar da olabilir. Ancak, herhangi bir tanımlayıcı belge bulunmadığı için kaynak kod bir gizem içindedir. Geliştirildiği sırada hakim olunan yazılım, üzerinden yıllar geçince ve geliştiricilerden kimse bulunamayınca bakım işi tam bir karmaşaya döner. Böyle durumlarda uygulanan tersine mühendislik, yazılımı kaynak koddan daha üst düzeyde bir soyutlama ile yeniden tanımlamaya çalışma sürecidir. Bu aslında tasarımın geri kazanılmasıdır. Bu maksatla geliştirilmiş olan çeşitli araçlar yardımıyla yazılım kaynak kodundan veri, mimari ve yordamsal tasarımı elde etmek mümkündür. Bundan sonra gerekli bakım işlemi yapılabilir.

Tersine mühendislikte amaç, kaynak kodu girdi olarak alıp tam tasarım belgelendirmesini çıktı olarak elde etmektir. Ancak, pratikte bunu böyle işlemesi her zaman mümkün olmaz. Sebebi, bu işi yapabilmek için gerekli olan yardımcı araçların sağlayacağı soyutlama düzeyinin kabul edilebilir olup olmamasıdır. Her araç kendi mantığına göre çalışmasını yaparak kaynak kodu belgeye çevirir. Bundan sonrası yine yazılım mühendisleri tarafından incelenmeli, gerekirse elle düzeltmeler yapılmalıdır. Soyutlama düzeyinin yüksekliği, özellikle büyük yazılımlarda, tasarımın bakım mühendisleri tarafından daha iyi anlaşılmasını sağlar.

Tersine mühendisliğin uygulandığı bir başka alan da kaynak kodu hiç bulunmayan çalışır durumdaki programlardan tasarımı ele etmektir. Çalışan programların nesne kodu ikili durumda makine kodu içerirler. Günümüzde bu makine kodunu anlayıp okunabilir halde bir programlama dili haline çeviren araçlar da vardır. Ancak, modül, yordam ve değişken isimleri mantıksal deyimler içermeyebilir, yalnızca takibi kolaylaştıracak tanımlayıcılar içerir. Bu tanımlayıcıları daha sonra arzu edilen sözcüklerle değiştirmek mümkündür.

Her iki yöntemde de tersine mühendisliğin tam olması gerekir. Diğer bir deyişle, kodun hiçbir kısmı atlanmamalı, tasarım tam olarak kapsanmalıdır.