12. Hafta
1. Grup
Boşlayıcı

12.4. Hata Ayıklama (12.4.)

Bilgisayarlı sistemlerin en büyük sorunlarından biri hata durumunda hangi bileşenin kusurlu olduğunun belirlenmesidir. Kusur, donanımda olabileceği gibi, yazılımda da olabilir, hatta yalnızca basit bir kabloda olabilir. Kusurun yazılımda olduğunun bulunmasından sonra da hatalı çalışan yazılımların nerede hata yaptıklarını bulmak yeni bir sorun olarak ortaya çıkar. Ne yazık ki, günümüzdeki pek çok programlama dili ve derleyicisi yürütme anındaki hataların neden kaynaklandığını bize net olarak söyleyememektedir. Bu nedenle, kodlayıcılar, olası hataların nerelerde meydana gelebileceğini önceden kestirerek hata meydana geldiğinde yakalamak, cinsini, sebebini ve tam yerini raporlayarak belirli bir yere kaydettirmek durumundadırlar. Bunun için tüm sistem genelinde bir hata raporlama ve veri toplama sistemi oluşturulmalıdır.

Hata ayıklama (debugging) aslında kod yazımına başlandığı zaman başlar. Kodlayıcı, yazdığı kodu sürekli okuyarak ilk hataları bulmaya çalışır; ondan sonra derleyicinin hata denetiçisini kullanır. Derlenip bağlanarak yürütülebilir hale getirilen yazılım biriminin iç yapısına bundan sonra yalnızca özel yazılım araçları ile ulaşılır. Bu araçlara *hata ayıklayıcı* (debugger) adı verilmektedir. Kullanılan programlama dili, işletim sistemi ve geliştirme ortamının özelliğine göre değişik yapıda hata ayıklayıcılar kullanılır. Hata ayıklayıcının kullanılamadığı yerlerde ekrana ya da bir dosyaya değişkenlerin değerlerini yazdırmak gibi klasik yöntemler kullanılır.

Test aşamasında herhangi bir hata bulunması aslında başarı olarak değerlendirilir. Bulunan hataların giderilmesi için de hata ayıklama süreci başlatılır.

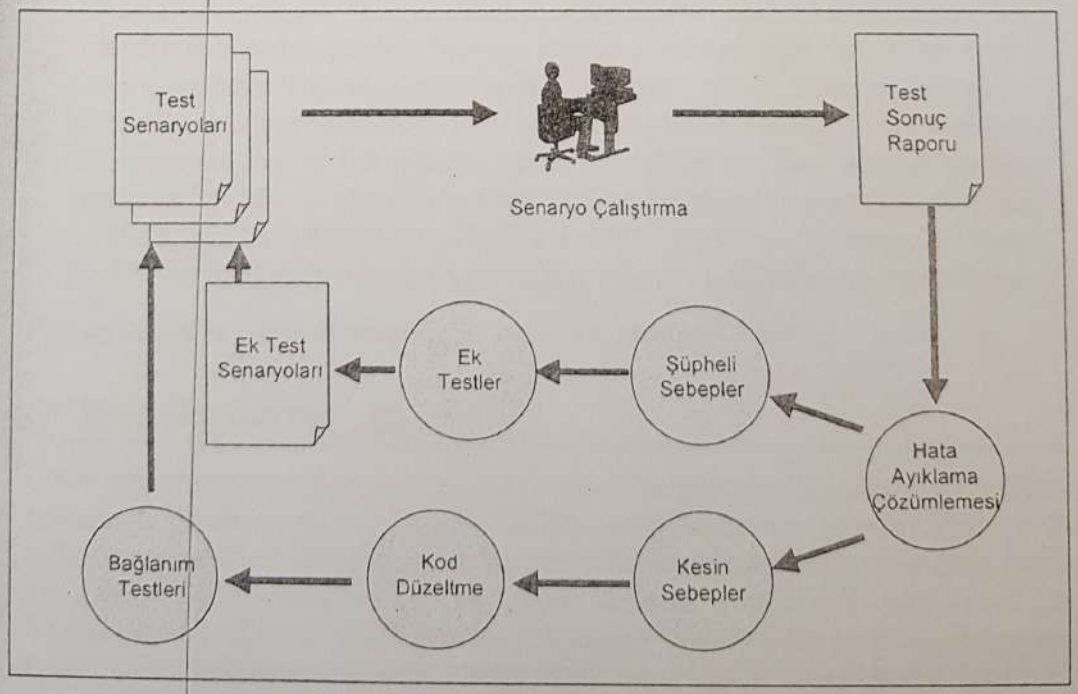
12.4.1. Hata Ayıklama Süreci

Test sırasında bulunan hatanın nereden kaynaklandığının belirlenmesi genellikle geliştiricinin deneyimine bağlıdır. Yeri ve sebebi hemen belirlenemeyen hatanın araştırılması için bir çözümleme yapılması zorunludur. Çoğu zaman, deneme-yanılma yöntemi kullanılarak hatanın giderilmesi bir rastlantıya bırakılır. Oysa bu iyi bir yöntem değildir. Çünkü her defasında aynı hatayı test ortamında yaratabilmeyi gerektirir. Buna ek olarak, test ortamının paralel çalışan gruplarca ortak kullanıldığı durumlarda, karmaşa yaşanabilir. Bu nedenle hatanın kaynaklandığı yeri belirleyebilmek için hemen kod üzerinde oynamalara girişmeden önce, tasarımdan kodlamaya kadar sağlam bir değerlendirme yapılması daha yararlıdır.

Hata ayıklama süreci normalde test aşamasının sonunda başlar. Testlerin mutlaka bir tek aşama şeklinde yapılma zorunluluğu olmadığı için, aslında test yapılabilecek her noktada hata ayıklama vardır. İlk kodlama anında ortaya çıkarılan hatalardan sistem kullanımı sırasında rapor edilen hatalara kadar her aşamada karşılaşılan yazılım hataları için birer hata ayıklama süreci uygulanır. Bu sürecin amacı, yazılımın kusurlu çalışan kısmının neden kaynaklandığını ipuçlarını değerlendirerek bulmayı ve gidermeyi içerir. İpuçlarının veya hataya neden olduğu kanısına varılan belirtilerin takip edilmesiyle ya başarıya ulaşılır, yani hatalı çalışan kısım bulunur ve düzeltilir, ya da başarısız olur, yani hata bulunamaz. Başarısız olunan durumda, hata ayıklayıcı kişi veya kodlayıcı, test senaryoları yazarak kodun belirli kısımlarını test eder, gerekirse tekrarlar halinde hatayı arar.

Yaygın olarak kullanılan bir hata ayıklama süreci Şekil-8.6 da gösterilmektedir. Geliştirilen yazılımın test edilmesi için hazırlanan test senaryoları çalıştırılarak test sonuç raporu üretilir. Bu raporda yer alan yazılım hatalarını düzeltebilmek üzere hata ayıklama çözümlemesi yapılır. Bu çözümleme sonunda hatanın kesin sebebi bulunursa, kodun kusurlu yerleri düzeltilir, bu düzeltmelerden etkilenmesi olası kısımlar için *bağlanım* yani regresyon (regression) testleri yapılır, test senaryoları tekrar çalıştırılarak test sonuçları değerlendirilir.

Çözümleme sırasında kesin ol-duklarından emin olunamayan şüpheli hata nedenleri için ek testler tasarlanır, yeni senaryolar hazırlanır. Bu senaryolar çalıştırılarak yeni test sonuçları elde edilir ve tekrar çözümleme yapılır. Bu süreç test sonuçları başarılı bulununcaya kadar devam eder [5].



Şekil-8.6. Hata ayıklama süreci.

8.4.2. Yaygın Hataların Özellikleri

Yazılım mühendislerinin ve kodlayıcıların dikkate alması gereken başlıca hatalar, hata belirtileri ve olası önlemler şunlardır:

- Yazılımın kusurlu çalışmasına neden olan hata durumu bir modülde iken onun belirtisi bir başka modülde olabilir; bu sıkı geçmeli programlamada oldukça yaygındır. Hatanın belirlenebilmesi için işlevsel çözümleme iyi yapılmalıdır.
- Bir hata düzeltildiğinde, başka bir hatanın belirtisi ortadan kalkmış olabilir. Bu durumda, diğer hatanın nedeninin de aynı olduğu kanısına varılabileceği gibi, değişik girdilerde benzer hatanın oluşup oluşmadığının kontrol edilmelidir.
- Hata belirtisi gerçek bir hataya ait olmayıp doğru çalışmanın yanlış çıktı vermesi şeklinde yorumlanmış olabilir. Hatanın kaynaklandığı yerde bir kusur bulunamamışsa, çıktıları (görsel, sesli, metinsel veya tetikleyici olay) sağlayan tüm yordamlar gözden geçirilerek doğru çalıştıkları sınanmalıdır.
- Belirti, bilgi işlemeden değil bir zamanlamadan kaynaklanmış olabilir. Zamana bağlı olarak işlem yapan yazılımlar, özellikle gerçek zamanlı yazılımların koşutzamanlı çalışmadaki uyumsuzlukları hataya yol açabilir. Bu hataları bulabilmek oldukça zor olduğundan, zamana bağlı her işlev önce kendi başına sınanmalı, zaman kısıtlarını aşmamak kaydıyla, belirli işlevlere tahsis edilen zaman dilimlerinde bilinçli gevşetmeler yapılmalıdır.
- Hata belirtisi, kusurun yazılımdan ya da donanımdan kaynaklanmış olabileceği hakkında yeterince açık değildir. Bu durumda, yazılımın çalışmakta olan sürümünün daha önce doğru çalışıp çalışmadığının kaydı aranmalıdır. Bunun için de düzenleşim yönetiminin önemi ön plana çıkmaktadır. Bundan sonra da donanımın doğru çalıştığından, herhangi bir arızanın olmadığından emin olunmalıdır. Bağlantılar, kablo ve soketler karmaşık sistemlerde en çok hata üreten parçalardır. Önce bunların sağlam oldukları belirlenmeli, ondan sonra çeşitli kayıtların tutulduğu bir yazılım sürümü ile aynı hatayı yeniden yaratabilecek senaryolar oluşturulmalı, elde edilen verilere göre hatanın yeri saptanmalıdır.
- Dağıtık sistemlerde oluşan hatalar, yerleşke farklılığından doğmuş olabilir. Özellikle iletişim hatalarından kaynaklanan koşutzamanlılık sorunları farklı düğümlerde çalışan yazılım birimleri arasında veri iletişimini ve bilgi işlemeyi etkileyerek kusurlar yaratabilir. Bu nedenle arayüzler ve iletişim yolları çok iyi sınanmalıdır. Bazı durumlarda, sistemin çalıştığı ortam elektromanyetik yayınlardan fazlasıyla etkilenebilir. Bu da iletişim yollarında hiç beklenmedik hatalara yol açabilir. Uygun kablolama ve veri aktarım protokolleri kullanılarak bu tür hataların önüne geçilmeli ve çok dikkatli test edilmelidir.

8.4.3. Hata Ayıklama Yöntemleri

Hata ayıklamanın tek amacı yazılım hatasına neden olan kusuru bulmak ve düzeltmektir. Bu amacı gerçekleştirmek sistemli bir çözümleme, değerlendirme, deneyime ve mantığa dayalı tahmin ve ne yazık ki biraz da şans ile mümkün olur. Biz şimdi yaygın olan birkaç hata ayıklama yöntemini görelim:

- **Kaba kuvvet**
  Kaba kuvvetle (brute force) hata ayıklama yönteminde, yürütme anındaki davranışlar izlenir, yazılım biriminin çeşitli noktalarına ekrana veya bir dosyaya o an akışın neresinde olduğunu, genel durumu veya bir değişkenin değerini yazan deyimler eklenir. Bu bilgiler ışığında, hataya neden olan yazılım kusuru aranır. Bazen hemen sonuç verir, bazen de çok miktarda kod yazılması nedeniyle aşırı zaman ve emek kaybına yol açar. Öte yandan, bu yöntem çoğunlukla başarıya ulaşır. En çok kullanılan yöntem olmasına karşın kaba kuvvet yöntemi aslında verimli değildir. Bir ilke olarak, diğer yöntemler denenip de sonuç alınmadığı zaman ya da kesin sonuç alınacağından emin olunduğunda uygulanmalıdır.

- **Geri izleme**
  Özellikle küçük yazılımlarda yaygın olarak kullanılan bir yöntem olan geri izleme (backtracking) kodun okunarak izlenmesi esasına dayanır. Hatanın oluştuğu yerden itibaren geriye doğru gidilerek kod incelenir; hata yaratan deyim ya da kusurlu akış mantığı aranır. Yüksek satır ve modül sayısına sahip yazılımlarda, akış yolunun çok dallandığı noktalarda bunu yapmak oldukça zordur.

- **Neden eleme**
  Tümevarım (induction) veya tümdengelim (deduction) yöntemlerine dayanarak elde edilen verilere göre hatanın nedeni araştırılır. Ortaya konan varsayımları doğrulayıcı ya da çürütücü testler tasarlanır. İlk testler olumlu sonuç verirse, daha ayrıntılı verilerle testlere devam edilerek hatanın tam yerinin saptanmasına çalışılır.

- **Yardımcı araçlar**
  Günümüzdeki pek çok derleyici, hata ayıklayıcılarla beraber kullanıma sunulmaktadır. Bunlardan bazıları derleme anında yazılımın olası hatalarını incelerken (uygunsuz atamalar, olası bellek sızıntıları, yanlış adresler, iyileştirilebilecek noktaların belirtilmesi gibi) bazıları da yürütme anında yazılımın davranışlarını izlerler ve durumlarını sürekli kaydederler. Bazı araçlar verilen yazılımı adım adım yürütebilirler (tracer). Bazıları da otomatik test senaryosu üretebilir, yazılım çöktüğü anda bellek içeriğini kaydeder ve sonradan inceleyebilme olanağı sağlar, simge veya çapraz referans tablolarını oluşturabilirler.

Hata ayıklamada pratik öneriler Ayrıt 14.3.7 de verilmektedir.

12.5. Belgelendirme (12.5.)

Çözümleme ve tasarım evrelerinde düzenli olarak belgelendirme kullanıldığı gibi test aşamasında da bir plana göre yapılan testlerin belgelendirmesi yapılır. Sistemin bir bütün olarak nasıl test edileceği, sistemin doğrulamasının ne şekilde yapılacağı, bunun için gerekli olan yardımcı araçlar bir planda belirtilir. Bu belgenin adı *Sistem Test Planı* (System Test Plan)'dır. Bu planda, donanım ve yazılım için ne tür bir test ortamı kullanılacağı, testlerin tanımlaması, nasıl yapılacağı anlatılır ve bir takvim verilir. Bu plan için bir şablon Ek-A.11 de verilmektedir.

Genel sistem isterleri doğrultusunda hazırlanan sistem test yordamları ile Test Planı'na göre testler yapılır. Büyük sistemlerde ya da salt yazılımdan oluşan sistemlerde, ana bileşenlerinden biri olan yazılımın testinin nasıl yapılacağı bir *Yazılım Test Planı* (Software Test Plan) içinde açıklanabilir. Bu planda, yazılım tümleştirmesinin ve testinin nasıl yapılacağı, ne tür tümleştirme stratejisi kullanılacağı, geliştirme süreci içindeki testle ilgili planlamalar, test evreleri ve zaman çizelgeleri belirtilir. Evrelerin başlangıç ve bitiş tarihleri, bu evrelerde test edilecek yazılım birimleri ve kümeler tanımlanır. Ek yük getirmesi olası geçici kod parçalarının özellikleri, test için kullanılacak donanımın, benzetim ortamlarının ve benzeticilerin özellikleri, özel test gereçleri ve kullanım teknikleri, kullanıcıya gösteri amacıyla yapılan kabul testlerinin nasıl yapılacağı bu belgede belirtilir. Yazılım Test Planı belgesinin ana çatısı Ek-A.21 de verilmektedir.

Testlerin yapılış şekilleri, test senaryoları ve beklenen sonuçlar *Yazılım Test Tanımlaması* (Software Test Description) belgesinde anlatılır. Çok büyük yazılımlar daha küçük yazılım öğelerine bölünerek gerçekleştirim, tümleştirme ve test yapıldığından birden fazla Test Tanımlaması bulunabilir. Herbir tanımlamada, hangi isterlerin ne şekilde test edileceğinin belirtilmesi yanında, arayüz bütünlüğü, işlevsel doğruluk, başarım, yükleme gibi özellikleri sınamak üzere test yordamları tanımlanır. Herbir test yordamı aşağıdaki öğeleri içerir:

- Testin tanımı ve amacı
- Sistemin içinde bulunması gereken durum çalışma kipi
- Sıralı liste halinde girdi olarak uygulanacak etkinin tanımı ve veri girdisinin değeri
- Sıralı liste olarak herbir etkinin karşılığında alınacak sonucun tanımı ve veri çıktısının değeri

Yazılım Test Tanımlaması'nın içeriği Ek-A.22 de, *Yazılım Test Sonuç Raporu* (Software Test Result Report) şablonu da Ek-A.23 te verilmiştir. Sonuç raporunda, her testin geçerli olup olmadığı, karşılaşılan sorunlar, eklenmesi ve düzeltilmesi istenen konulardan oluşan bir istek listesi yer alır.

Yazılım bir müşterinin siparişine göre geliştirilmişse, yapılan sözleşmeye bağlı olarak, bir *Yazılım Kabul Testi Raporu* (Software Acceptance Test Report) düzenlenir. Bu raporun hem geliştirici hem de müşteri tarafından onaylanması ile yazılımın kesin teslimi ve kabulü yapılmış olur.

Bu belgeler yazılım geliştirme süreci içinde hazırlanır ve üretilen yazılımın testi sonuçlandıktan sonra düzenleşim yönetimine teslim edilir.

12.6. Riskler

İyi hazırlanılmadan başlanan bir testin önce kendisi büyük bir risk oluşturur. Normal bir süreç olarak izlenen yazılımın test aşamasında da karşılaşılması olası temel riskler şunlardır:

- *Tümleştirmede karmaşa*: Tek bir birimden oluşan yazılımın testi başka birimleri etkilemeyeceği için bir karmaşıklık oluşmaz. Ancak, birden fazla yazılım biriminin tümleştirilmesi ve testi sırasında karşılıklı olumsuz etkilenmeler oluşabilir. Aynı anda test edilen birim sayısı arttıkça hataların nederen kaynaklandığı bulmak da güçleşir.
- *Testlerin plan dışı yürütülmesi*: Testlerin bir plana göre yürütülmemesi durumunda test ortamının kullanımında, testlerin uygulanmasında, hata bulmada karmaşa yaşanabilir. Plansız yaplan testlerle zaman kaybı yaşanabilir, hataların kaynaklanma nedeninin bulunması zorlaşır.
- *Test yordamlarının yetersizliği*: Test yordamları yeterince kapsamlı olmayıp yüzeysel olarak uygulanırsa testler başarılı geçse bile atlanmış durumların gerçek kullanım sırasında sorun çıkarması beklenmelidir.
- *Sıralama*: Test yordamlarının belirli bir sıra takip etmeleri gereklidir. Uygun bir sıraya göre yapılmayan ve senaryosu iyi tanımlanmayan test durumları tekrarlara ve sonuçta da zaman kaybına yol açabilir.
- *Paralel test işlemleri*: Birden fazla öğe aynı anda test ediliyorsa ve her bir öğe içinde birden fazla birim bulunuyorsa, yürütme sırasında birbirlerine olan etkilerinden dolayı hangi öğenin doğru, hangisinin kusurlu olduğunun bulunmasında güçlük çıkabilir.
- *Yüksek maliyetli testler*: Bazı testler çok yüksek maliyet gerektirebilir. O nedenle son derece dikkatle planlanmalı, önceden yeterli bütçe ayrılmalı, test sonrasında iyi değerlendirme yapılabilmesi için kayıtlara önem verilmelidir.
- *Bazı testlerin yapılamaması*: Can ve mal kaybını önlemek üzere gerçekleştirilmiş sistemlerin, özellikle askeri savunma sistemlerinin bir kısım testlerini yapmak hiçbir zaman mümkün olmayabilir. Böyle testler ancak benzetim ortamlarında yapılabilir veya bir savaş durumunda denenebilir.

12.7. Sistem Teslimi

Sistemin testlerden başarıyla geçip kabul edilmesinden sonra sistemin bakım ve onarım konularında gerekli belgeler eğitimlerle birlikte kullanıcıya teslim edilir.

12.7.1 Eğitim

Sistem teslim edilmeden bir süre önce sistemi kullanacak olan personele eğitim verilmelidir. Bu eğitim aynı zamanda büyük sistemlerin testleri sırasında personelden yararlanılabilmesi için de gereklidir.

Kuramsal eğitim herhangi bir derslikte sistem kullanıcı rehberleri üzerinden verilebilir. Bundan sonra çalışan bir sistemin kullanıcı arayüzü üzerinde uygulamalı eğitim yaptırılır. Salt yazılımdan oluşan ve ticari donanım kullanan sistemler için eğitim herhangi bir yerde verilebilir.

Eğitimin son aşaması kullanıcılar iş başında iken verilir. Sistem çalışacağı yere kurulduktan sonra gerçek verilerle, gerçek koşullarda kullanılmaya başlanır. Bu arada kullanıcıların karşılaşabilecekleri sorunları gidermek, anlayamadıkları kısımlarda kendilerine yardımcı olmak üzere bir süre destek sağlanır. Bu yöntem ancak kullanım ortamının gizlilik durumu elverdiği takdirde uygulanabilir.

Donanım içeren sistemler için ayrıca donanım bakım eğitimi verilir. Bu eğitimde, sisteme nasıl düzenli bakım uygulanacağı, bir arıza olması durumunda ne tür arıza bulma yöntemleri izleneceği, arızalanan birimlerin nasıl değiştirileceği ve onarımların hangi düzeyde, nasıl yapılacağı kapsanır.

Çok sayıda dağıtımı yapılan paket yazılım ürünleri için tanıtıcı düzeyde ve ileri düzeyde kullanıcılar için eğitimler verilebilir.

Destek (12.7.2)

Normal olarak bir geliştirici, piyasaya sürdüğü her ürün için teknik destek sağlamalıdır. Bu destek, yerinde eğitim olabileceği gibi, sanal ortamda eğitim, belge desteği ve çağrı merkezi şeklinde olabilir. Kullanıcı ilk kez bilgisayarlı bir ortama geçiyorsa *yerinde destek* adı verilen teknik yardım oldukça büyük önem kazanmaktadır. Özellikle bilgisayarı yeni öğrenen personel ile kullanılacak sistemlerin gerçekten verimli hale gelebilmesi için bir süre teknik destek ekibinin kullanıcı yerinde kalması, sorunlara çare bulması, kullanıcıların sorularını sabırla yanıtlaması ve sistemi öğretmesi gereklidir. Bu destek resmi anlamda bir eğitim gibi olmayabilir. Bu durumda, desteği alanın tam olarak ne istediğini bilmesi önemlidir.

Garanti (12.7.3)

Geliştirilen her sistem belirli bir süre garanti altına alınır. Bu süre içinde çıkan donanım arızaları ya ücretsiz olarak ya da düşük bir bedelle onarılır. Yazılım hataları ise genellikle ücretsiz olarak giderilir.

Özel geliştirilmiş sistemlerin donanım ve yazılımları sözleşmelerde tanımlı süre boyunca ücretsiz giderilir. Yazılımda, isterlerin yanlış anlatılmasından doğmuş kullanım eksikliği varsa, bu düzeltmenin garanti kapsamında mı yoksa değişiklik olarak mı algılanacağı yine ikili anlaşmalara bağlıdır. Bu kararın nasıl bir yöntemle verileceği konusunda baştan karar alınması sonradan anlaşmazlık çıkmasını önlemiş olur.

Garanti koşulları tamamen geliştirici ve müşteri arasındaki anlaşmaya bağlı olup belirgin bir kuralı yoktur. Anlaşmazlığın büyümesi halinde, sözleşmede kabul edilmiş hakemlere ya da mahkemelere gidilir.