11. Hft
1. Grup
Barlayıc

### 12.2.3. Yeterlilik Testleri

Yazılım *yeterlilik* (qualification) testi çoğu zaman *doğrulama* (verification) ve *geçerleme* (validation) konusunun bir parçası olarak değerlendirilir. Tümleştirme testinden sonra yazılım büyük ölçüde kusurlarından arındırılmış, arayüzleri doğru çalışır hale gelmiştir. Bundan sonra bir dizi test daha yapılır. İşte bunlar yeterlilik testleridir [1] [5]. Çoğu zaman bu testlerin ne olduğu sorusu akla gelir. En yaygın yanıt, müşterinin umduğu şekilde çalışmanın sağlanması ve bunun gösterilmesidir. Müşteriyi memnun edebilecek çalışma şekli Sistem İsterleri Belirtimi belgesinde tanımlanır. Sistem özelliklerini tanımlayan bu belirtim belgesi aynı zamanda geçerleme ilkelerini anlatan bir bölüme sahiptir. İşte bu bölümden test yordamları ve senaryolar türetilir. Benzer şekilde, yazılım isterleri de Yazılım İsterleri Belirtimi belgesinde tanımlanır.

Aşağıdaki standartlar yeterlilik testleri sırasındaki etkinlikleri, yapılması gereken işleri, doğrulama ve geçerleme sürecini tanımlamaktadır:

- IEEE 1012-1998, Standard for Software Verification and Validation
- IEEE 1059-1993, IEEE Guide for Software Verification and Validation Plans
- IEEE 12207.1, Guide for Information Technology - Software life Cycle Processes - Life Cycle Data

#### 12.2.3.1. Doğrulama

*Doğrulama* (verification), bir yazılım ürününün belirli işlevleri doğru olarak gerçekleştirdiğinden emin olmak için yapılanların tümüne verilen addır. Yazılım geliştirme süreci izlenirken belirlenen isterler arasında uyumsuzluk varsa, üst düzey isterden alt düzey isterin türetilmesi sırasında bir hata yapılmışsa, sonuçta gerçekleştirimi yapılan ürün asıl tanımlanan üründen farklı olabilir.

(iptal değil!)

Tümleştirme testleri genellikle yazılım geliştirenler tarafından değil de test için görevli ekiplerce tarafından kara kutu yöntemleriyle yapılır. Tabii ki test işlemleri belirli bir eşgüdüm içinde yürütülür.

Bazı geliştiriciler bir yazılım paketinin tüm birimlerini aynı anda bir araya getirerek test etme yöntemini seçerler. Daha önce iyice test edildiğinden emin olunan, kendini ispatlamış birimlerle bunu yapmak mümkün olabilir. Ancak, yeni geliştirilen birimlerden oluşan yazılım paketinin bütününün testi sırasında karşılaşılan kusurları düzeltmek son derece güçtür. Öncelikle sorunun hangi birimden kaynaklandığını belirlemek, sonra da o sorunun birim içindeki tam yerini ve nedenini bulmak gereklidir. Sorunu çözmek için yapılan işlemin başka birimleri de olumsuz şekilde etkilemesi mümkündür.

Bir anda tümleştirme yerine, yazılım paketini küçük parçalara bölerek birimler halinde artımlı olarak tümleştirmek daha sağlıklı sonuç verdiği için yaygın olarak kullanılan bir yöntemdir. Artımlı tümleştirme yönteminde değişik stratejiler kullanılabilir. Şimdi bunları görelim.

### 8.2.2.1. Yukarıdan Aşağı Tümleştirme

Yazılımı oluşturan birimlerin denetim sıradüzeni içinde, birimler ya *derinlik* tabanlı ya da *genişlik* tabanlı olarak tümleştirilirler. Önce ana denetim biriminin testi yapılır, sonra ona en yakın düzeydeki birimlerden biri ile beraber test yapılır. Bir sonraki aşamada ya aynı düzeydeki bir başka birimin testi yapılır; ya da aynı birimin bir alt düzeyindeki birimle tümleştirmesi ve testi yapılır. Bu işlemler sırasında, tümleştirmesi daha sonra yapılacak olan birimin geçici olarak yerine geçecek şekilde basit kod parçaları veya sürücüler kullanılır. Test sırası geldikçe bu geçici kod parçaları gerçekleriyle değiştirilir. Herbir eklemede daha önce yapılan testlerin bir kısmı ya da tamamı tekrar yapılarak yeni bir kusur eklenip eklenmediğinden emin olunur.

Bu test stratejisi, yazılımda denetim ve akış sorunları varsa bunların öncelikli olarak belirlenmesini, tasarımda öngörülen önemli karar verme noktalarının erkenden gerçeklenmesini sağlar. Eğer derinlik tabanlı yöntem kullanılırsa, bir işlevin tamamının gerçekleştirilmesi ve test edilmesi sağlanabilir.

Yukarıdan aşağı tümleştirme stratejisinin en büyük sorunu geçici yazılım birimlerinin (stub) uygun şekilde hazırlanmasıdır. Gerekli tüm testleri yapabilmek için oldukça yetenekli kod parçalarının geliştirilmesi gereklidir. Bu nedenle, ya yeterli veri giriş/çıkışını sağlayabilecek kod geliştirilmeli ya da ayrıntılı testler sonraya ertelenmelidir.

### 8.2.2.2. Aşağıdan Yukarıya Tümleştirme

Bu stratejide tümleştirme atomik birimlerin çalıştırılıp test edilmesiyle başlar. Alt düzey birimler birleştirilerek kümeler haline getirilir. Bu küme, test senaryosu gereğince geliştirilen bir sürücü ana programla veri giriş/çıkışı bakımından test edilir. Benzer şekilde, diğer alt düzey birimler de kümeler halinde test edilir. Daha sonra sürücü programlar ayrılarak yazılım paketinin yapısı içinde bulunan bu kümelerin

birleştirilmesinden oluşan daha üst düzeyde yeni kümeler meydana getirilir. Bu kümeler de yine sürücü programlarla test edilir. Bu şekilde en üstte bulunan ana birime kadar ulaşılır. Tümleştirme üst düzeylere doğru çıktıkça test sürücülerine olan gereksinim azalır. Bu stratejinin en büyük sorunu da tümleştirme sonuçlanmadan önce ana yazılım paketini oluşturmanın mümkün olmamasıdır.

Bazı dağıtık sistem yazılımları büyük kümelerden oluşan yazılım öğeleri şeklinde geliştirilir ve test edilirler. Herbir yazılım öğesi ya bütün olarak ya da aşağıdan yukarı yöntemle tümleştirilir ve test edilir.

Hangi stratejinin seçileceği yazılımın özelliğine, yazılım geliştirme deneyimlerine ve bazen de proje planlamasına bağlıdır. Her iki stratejinin üstünlüklerinden yararlanabilmek için üst düzeydeki birimlerin yukarıdan aşağı, alt düzeydeki birimlerin de aşağıdan yukarı tümleştirilmesi en yararlı yöntem olarak benimsenmektedir. Testler sırasında en kritik olarak değerlendirilen birimlere özel önem verilerek test senaryolarının onları iyice kapsaması sağlanmalıdır.

### 12.2.3. Yeterlilik Testleri

Yazılım *yeterlilik* (qualification) testi çoğu zaman *doğrulama* (verification) ve *geçerleme* (validation) konusunun bir parçası olarak değerlendirilir. Tümleştirme testinden sonra yazılım büyük ölçüde kusurlarından arındırılmış, arayüzleri doğru çalışır hale gelmiştir. Bundan sonra bir dizi test daha yapılır. İşte bunlar yeterlilik testleridir [1] [5]. Çoğu zaman bu testlerin ne olduğu sorusu akla gelir. En yaygın yanıt, müşterinin umduğu şekilde çalışmanın sağlanması ve bunun gösterilmesidir. Müşteriyi memnun edebilecek çalışma şekli Sistem İsterleri Belirtimi belgesinde tanımlanır. Sistem özelliklerini tanımlayan bu belirtim belgesi aynı zamanda geçerleme ilkelerini anlatan bir bölüme sahiptir. İşte bu bölümden test yordamları ve senaryolar türetilir. Benzer şekilde, yazılım isterleri de Yazılım İsterleri Belirtimi belgesinde tanımlanır.

Aşağıdaki standartlar yeterlilik testleri sırasındaki etkinlikleri, yapılması gereken işleri, doğrulama ve geçerleme sürecini tanımlamaktadır:

- IEEE 1012-1998, Standard for Software Verification and Validation
- IEEE 1059-1993, IEEE Guide for Software Verification and Validation Plans
- IEEE 12207.1, Guide for Information Technology - Software life Cycle Processes - Life Cycle Data

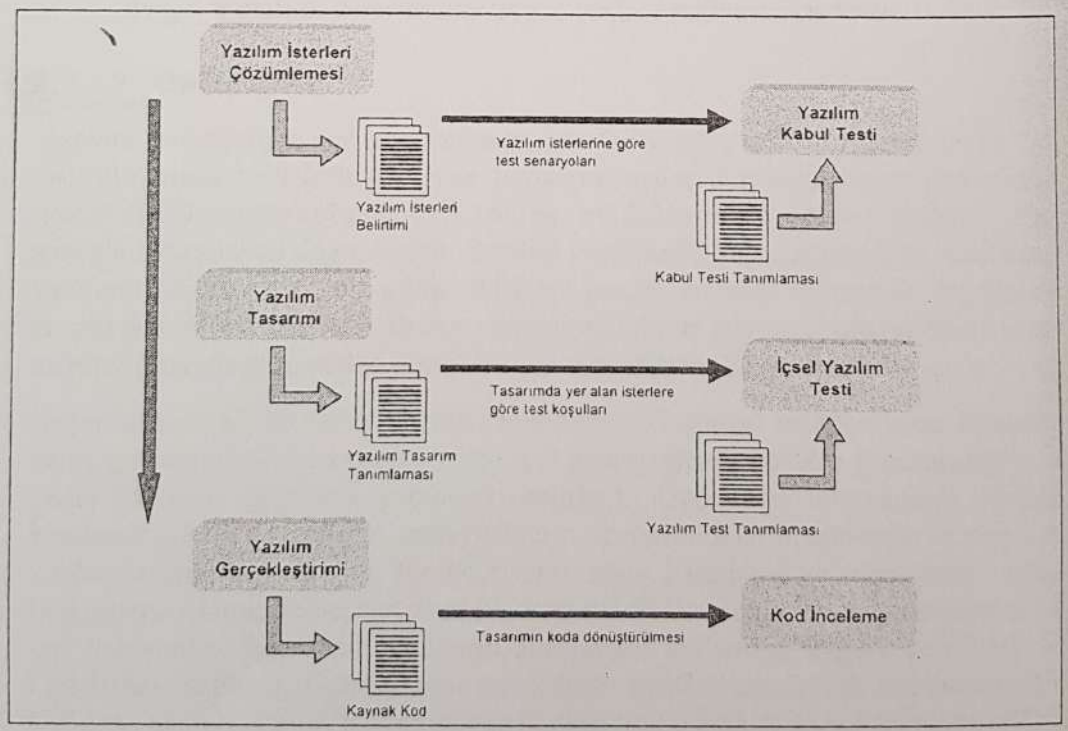
#### 12.2.3.1. Doğrulama

*Doğrulama* (verification), bir yazılım ürününün belirli işlevleri doğru olarak gerçekleştirdiğinden emin olmak için yapılanların tümüne verilen addır. Yazılım geliştirme süreci izlenirken belirlenen isterler arasında uyumsuzluk varsa, üst düzey isterden alt düzey isterin türetilmesi sırasında bir hata yapılmışsa, sonuçta gerçekleştirimi yapılan ürün asıl tanımlanan üründen farklı olabilir.

Doğrulama testi yazılım isterleri çözümlemesi evresinde ortaya konan isterlerin, gerçekleştirimi tamamlanmış yazılım tarafından tam olarak karşılanıp karşılanmadığını sınamak ve yazılımın bu isterlere uygun olarak geliştirildiğini kanıtlamak için yapılır. Daha da kısa olarak doğrulama, tanımlanmış olan ürünün doğru bir şekilde üretilip üretilmediğinin sınanması olarak tarif edilebilir.

Örneğin, yazılım isteri olarak, belirli bir işlemin tamamlanması için 2 saniye süre verilmiş iken, ayrıntılı tasarımda bu işlemi gerçekleştirmek üzere yapılması gereken üç ayrı işe birer saniye verilmiş ise, sonuçta bu işlem 2 değil, 3 saniyede tamamlanacaktır. Sistemin genel kabul testleri sırasında bu ayrıntının sınanması güç olabilir. İşte bunun için, kodlanan yazılımın tasarıma, tasarımın da isterlere uyduğunu sınamak gereklidir.

Doğrulama testleri için uygulanabilecek bir süreç Şekil-8.5 te gösterilmektedir. Mümkünse bağımsız bir grup tarafından her aşamada ayrı testler ve incelemeler yapılarak ürünün isterlere uyduğu kanıtlanmalıdır.




**Şekil-8.5. Doğrulama testleri.**

Doğrulama sürecinde aşağıdaki işler yapılmalıdır:

- **Sözleşme doğrulaması**
  Geliştirici ile müşteri arasında yapılan sözleşme aşağıdaki kıstaslara göre doğrulanmalıdır:
  - Geliştirici tüm isterleri karşılayabileceği güvenini vermektedir.
  - İsterler tutarlı olup kullanıcı gereksinimlerini kapsamaktadır.

- İsterlere yapılacak değişiklikleri ve ortaya çıkabilecek problemleri kontrol edebilmek üzere yordamlar öngörülmüştür.
- Taraflar arasında, sahiplik, garanti, telif hakları ve gizlilik gibi konuları da içerecek şekilde işbirliği ve temas noktaları için yordamlar öngörülmüştür.
- İsterlerle birlikte kabul kıstasları ve yordamları öngörülmüştür.

• **Süreç doğrulaması**

Süreç aşağıda listelenen kıstaslara göre doğrulanmalıdır:

- Proje planlamaları yeterli ve takvime uygundur.
- Proje için seçilen süreçler yeterlidir, planlandığı şekilde yürütülmektedir ve sözleşmeye uygundur.
- Proje süreçleri için seçilmiş standartlar, yordamlar ve ortamlar yeterlidir.
- Proje personeli sağlanmış ve sözleşme gereklerine göre eğitilmiştir.

• **İsterler doğrulaması**

İsterler aşağıda belirtilen kıstaslara göre doğrulanmalıdır:

- Sistem isterleri tutarlı, gerçekleştirilebilir ve test edilebilir durumdadırlar.
- Sistem isterleri tasarım ölçütlerine uygun şekilde donanım öğelerine, yazılım öğelerine ve el işlemlerine atanmıştır.
- Yazılım isterleri tutarlı, gerçekleştirilebilir ve test edilebilir durumda olup sistem isterlerine uymaktadır.
- Emniyet, güvenlik ve kritik durumlarla ilgili yazılım isterleri doğrudur.

• **Tasarım doğrulaması**

Tasarım aşağıda belirtilen kıstaslara göre doğrulanmalıdır:

- Tasarım, isterlere göre tutarlıdır ve izlenebilir durumdadır.
- Tasarım, olayların, girdilerin, arayüzlerin, mantık akışının uygun dizilişlerini, zaman ve büyüklük tahsislerini, hata tanımlarını, hataya dayanıklılığı ve geri kazanmayı gerçekleştirmektedir.
- Seçilen tasarım, isterlerden türetilebilmektedir.
- Tasarım, emniyet, güvenlik ve diğer kritik durumlarla ilgili isterleri gerçekleştirmektedir.

• **Kod doğrulaması**

Yazılan kaynak kod aşağıda belirtilen kıstaslara göre doğrulanmalıdır:

- Kod, tasarıma ve isterlere göre izlenebilir, türetilebilir ve test edilebilir.
- Kod, tasarım ve isterlere göre gerekirse yeniden yazılabilir.
- Kod, doğru ve kodlama standartlarına uygun durumdadır.
- Kod, olayların, girdilerin, arayüzlerin, mantık akışının uygun dizilişlerini, zaman ve büyüklük tahsislerini, hata tanımlarını, hataya dayanıklılığı ve geri kazanmayı gerçekleştirmektedir.
- Kod, emniyet, güvenlik ve diğer kritik durumlarla ilgili isterleri gerçekleştirmektedir.
- Kod, yazan kişiden başkası tarafından rahatça okunabilir, anlaşılabilir ve bakım uygulanabilir.

- **Tümleştirme doğrulaması**
  Tümleştirme aşağıda belirtilen kıstaslara göre doğrulanmalıdır:
  - Her yazılım öğesinin bileşenleri ve birimleri öğeyle tam ve doğru olarak tümleştirilmektedir.
  - Sistemin donanım öğeleri, yazılım öğeleri ve el işlemleri sistemler tam ve doğru olarak tümleştirilmektedir.
  - Tümleştirme işleri belirli bir tümleştirme planına uygun olarak yapılmaktadır.
- **Belgelendirme doğrulaması**
  Belgelendirme aşağıda belirtilen kıstaslara göre doğrulanmalıdır:
  - Belgelendirme yeterli, uygun, tam, anlaşılabilir ve tutarlıdır.
  - Belgelendirme hazırlıkları takvime uygundur; bir gecikme nedeni olmaz.
  - Belgelerin düzenleşim yönetimi, sürüm, baskı ve dağıtım denetimi belirlenmiş yordamlara göre yapılmaktadır.
  - Belgeler gizlilik derecelerine uygun şekilde saklanmaktadırlar.

### 12.2.3.2. Geçerleme

Yazılım belirtiminde yer alan kullanıcı isterlerine göre ortaya mükemmel bir ürün çıkabilir, ancak bu ürünün gerçekten kullanıma uygun olup olmadığının da sınanması gereklidir. *Geçerleme* (validation), yani *geçerli kılma* ya da *sağlama*, kullanıcı gereksinimleri karşılayan doğru ürünün üretilip üretilmediğinin sınanmasıdır. Geçerleme, yazılımın doğru amaç için geliştirildiğinin gerçek ortamda denenerek doğruluğuna kanaat getirilmesi demektir. Bunun yanında, yazılımın işlevleri kadar iç özelliklerinin ve niteliğinin de arzu edilen düzeyde olup olmadığının denetlenmesini içerir.

Geçerleme süreci iki etkinlik içerir: Süreç gerçekleştirimi ve geçerleme. Bunlardan süreç gerçekleştirimi sırasında, proje için geçerlemenin gerekip gerekmediği ve ne kadar bağımsız gruplarca yapılacağı belirlenir. Geçerleme uygulanacak öğelerin, yapılacak işlerin ve testlerin, özkaynakların, sorumlulukların, takvimin ve raporlama yordamlarının tanımlandığı bir geçerleme planı hazırlanır ve uygulamaya konur. Doğrulama işleminde ise, test isterleri, test senaryoları, sonuçları değerlendirmek için test belirtimleri hazırlanır. Bunların amaçlanan kullanıma uygun, belirli isterleri karşılaması sağlanır. Germe, sınır ve uçdeğer testleri, hatalardan korunma ve geri kazanma testleri yapılır. Kullanıcı temsilcileri ürünü amaçladıkları kullanım şekliyle test ederler. Yazılımın amaca uygunluğu onaylanır, hedef sistem üzerinde, seçilen gerçek yerlerde kullanılarak sınanır.

Doğrulama için verdiğimiz örnekte, yazılım isterleri belirtiminde yer alan belirli bir işlemin tamamlanması için gerekli sürenin 2 saniye olarak tanımlandığını belirtmiştik. Gerçek ortam testleri sırasında, bu işlemin sonucunu beklemekte olan bir başka sisteme olan bağımlılıktan dolayı işlemin aslında 1 saniyede tamamlanmasının gerektiği anlaşıldığında, bu ister geçerliliğini yitirir, yani doğruluğunu kaybeder. Bu durumda, yazılımın yeni bir çözümleme ile tekrar gözden geçirilmesi gerekmektedir.

Bir başka örnek olarak, sistemin kullanılabilirliğinin 24 saat ilkesine dayandığı özel bir ister olarak ortaya konmuş iken, tasarımda bu isterin nasıl yerine getirildiği açıkça yer almamışsa, geçerleme için bütünlüğü bozan bu durumun düzeltilmesi ve öngörülen tedbirlerin sınanması gereklidir.

Geçerleme testi, geliştirilen sistemin kullanıcıya teslimi sırasında önem kazanır. Eğer kullanıcı isteklerinin tamamı karşılanmazsa, yani müşteri sonuçtan memnun olmazsa, yönetsel sorunlar ortaya çıkar. Ürünün çok nitelikli bir şekilde geliştirilmiş olması, yani doğrulanmış olması, onu geçerli kılmaz. Özellikle, geliştirme süreci sırasında müşterinin gözden geçirmelere katılmamış olması bu tür sorunlar doğurabilir.

Genellikle her test için üç sonuçtan biri elde edilir: Başarı, kısmen başarı veya başarısızlık. Tabii ki başarı durumu hem geliştirici hem de müşteri için arzu edilen sonuçtur. Kısmen başarı, belirlenen kusurların kısa sürede giderilebilecek cinsten olması durumunda geçerlidir. Bunların düzeltilmesi için müşteri ile geliştirici arasında karşılıklı anlayışla çözüm sağlanabilir. Fakat, başarısızlık durumunda hedeflenen zaman içinde kusuru gidermek mümkün değildir, ancak müşteri ile ek anlaşma yapılarak kusurun giderilebilmesi için çözüm aranır.

Geçerleme testleri özellikle görev-kritik sistemler için son derece önemlidir. Örneğin, askeri amaçlı bir füze savunma sisteminin doğrulama testleri laboratuvar ortamındaki bir test sistemi üzerinde yapılabilir. Ancak, geçerleme testlerini yapmak, o sistemi en gerçekçi koşullarda denemekle mümkün olur. Bu da sistemi gerçek çalışma ortamına çıkarıp gerçek bir füze atışına karşı tepkilerinin ölçülmesiyle yapılabilir. Böyle bir sistem için tek geçerleme koşulu vardır: Atılmış olan füzenin isabet etmeden önce füze-savar sistemiyle yok edilmesidir. Bu koşulun karşılanamaması ya da kısmen karşılanması kabul edilemez; tam sonuç alınması gereklidir. Böyle bir sistemin testi için gerekirse gerçek bir füze atışı dahi yapılmalıdır.

### 12.2.3.3. Rasgele Testler

Belirli bir test planına göre ve test yordamları uygulanarak yapılan testler doğru sonuç vermelerine rağmen, bazı olağan dışı durumlar oluştuğu takdirde sistemin nasıl davranacağı belirsiz olabilir. Örneğin, çok miktarda kullanıcı girdisi gerektiren bir yazılım, tüm işlevleri normal yapıyor olsa da, oluşması son derece düşük bir olasılığa sahip tuş kombinasyonlarından birinde hataya neden olabilir. Kullanıcının belirli bir sırayı takip ederek veri girişi yapması beklenirken, bir kullanıcının değişik bir sıra takip etmesi sistemi olumsuz etkileyebilir. İşte böyle önceden tahmin edilemeyen durumları gerçekleştirebilmek için yazılım test personeli veya kullanıcılar sistem üzerinde rasgele hareketlerle testler yapabilirler.

Bu tür testler bazı gruplarca, bilinçsiz hareketler yaparak bilgisayar kullanmaya çalışan bir maymundan esinlenilerek *maymun testi* (monkey test) olarak da adlandırılmaktadır. Yordamları resmi olarak tanımlanmış testlerden başarıyla geçen pek çok sistem bu tür testlerde başarısızlığa uğrayabilir. Ancak, önemli olan, ortaya çıkması olası her hatayı sistemi kullanıma vermeden önce yakalamak ve gidermektir.

### 12.2.4. Sistem Testi

Yazılım hiçbir zaman kendi başına bir sistem değildir; sistemin ancak bir parçasıdır. Her yazılım, belirli bir donanım üzerinde çalışır, bir takım altsistemlerle tümleştirilir ve topladığı bilgileri işleyerek sonuçlarını ya sergiler ya da doğrudan bir donanıma kumanda eder. Bu gerçekten hareketle, bilgisayar tabanlı bir sistemin doğru çalıştığını kullanıcıya göstermek üzere bir dizi test yapılır. Hem yazılım hem de donanımı kapsayan bu testlerin öncesinde de hata bulma ve sistemi deneme amacıyla hazırlık testleri yapılır. Testlerin şekilleri uygulama alanına göre değişiklik gösterse de hem donanım hem de yazılım bileşenlerini kapsayan sistem testleri, tümleştirme ve doğrulama amaçlarıyla yapılır. Bu testler, yazılım geliştirme sürecinin biraz dışında kalsa da, tasarım ve gerçekleştirimin iyi yapılması, özellikle büyük sistemlerin tümleştirme testlerinde daha az hatayla karşılaşılmasını sağlar.

Sistem testleri sırasında karşılaşılan en büyük sorun, bir hata ortaya çıktığında sistemi oluşturan bileşenlerin üreticilerinin birbirlerini kusurlu üretimle suçlamaları, hatanın kendilerinde olmadığını savunmalarıdır. Yazılım geliştirici, kendi paketlerinin geliştirme ortamında doğru çalıştığını savunurken, donanım bileşenlerinden birinin üreticisi, diğer bileşenin kendi çalışmasını etkilediğini iddia edebilir. Bu tür tartışmalar basit bir soketin yerine iyi oturmamış olmasının belirlenmesiyle sonlanabileceği gibi karşılıklı suçlamalarla da devam edebilir. Oysa kısır tartışmalar yerine yapılması gereken, donanım mühendisi için tüm bağlantıları ve donanım bileşenlerini kontrol etmek; yazılım mühendisi için de, tüm üretici ve geliştiricilerle beraber yapılacak test planlamalarına ve test tasarımlarına katılmak, potansiyel arayüz problemlerini araştırarak diğer bileşenlerden gelen verilerin akışını takip edip hata bulucu test tasarımları yapmak, arayüzlerden gelebilecek bozuk verileri yakalayıp rapor edebilecek düzeneklerle testler yapmak, test sonuçlarını kaydederek diğer hataların bulunmasına yardımcı olmaktır. Böyle durumlarda gerekirse sistem mühendisleri de devreye girebilir.

Bilgisayar tabanlı sistemlerin kullanımına başlamadan donanım ve yazılım testlerine ek olarak, sistem düzeyinde çeşitli testler yapılmalıdır. Şimdi bu testlerin neler olabileceğini görelim:

#### 12.2.4.1. Yükleme Testi

Her bilgisayar sistemi belirli tür ve miktarda veriyi işlemek ve bunlara göre başka veriler üretmek üzere tasarlanır. Gerçek zamanlı sistemler ve kontrol sistemleri daha çok belirli kesmelere karşı bir işlem yaparak belirli bir tepkide bulunurlar. veritabanı yönetim sistemleri ise çok miktarda veriyi saklama, bunlar üzerinde sorgulama yapma ve rapor üretme gibi işlevler yürütür. İşte bu tür yoğun veri akışına sahip sistemler için *yükleme testleri* (load test) yapılır. Tipik bir sistemin yükleme testleri şunlardan oluşabilir:

- Veri hacmi testi (volume test): Sistemin yüksek miktarda veri ile yüklenmesi
- Veri debisi testi (stress test): Tüm girişlerin yüksek hızda veri ile yüklenmesi
- Kapasite testi (storage test): Sistemin bellek ve disk kullanımının zorlanması.

Bu testler, sistemin sınırlarını zorlayarak en fazla ne kadar veri işleme kapasitesi olduğunu belirlemek, bu yükte davranışlarını kontrol etmek amacıyla yapılır. Hatta, bazı durumlarda, isterlerde belirtilen değerlerin de üzerine çıkılarak kaldırılabilecek en fazla yükün ne olabileceği araştırılır.

### 12.2.4.2. Germe Testi (12.2.4.2)

Yazılım testlerinde kullanılan çeşitli tekniklerle normal işlevler ve başarım sınanır. Ancak bu testler her şeyin normal olduğu durumlarda, en ideal koşullarda yapılır. Bir yazılım bileşeni tek başına test edildiğinde normal çalışıyorken, tümleştirildiği sistemin diğer bileşenleriyle beraber normal çalışması mümkün olmayabilir. Bu kusurun bulunması ve düzeltilmesi tümleştirme sırasında yapılabilir. Bir de normal olmayan koşullar oluşturulduğunda, hem yazılım hem de donanımın ne şekilde davranacağını görmek üzere *germe testleri* (stress test) yapılmalıdır. Bu amaçla yapılabilecek testler şunlardır:

- Sistemi normal olmayan miktarda özkaynak gerektirecek şekilde zorlamak amacıyla yapılan testler
  - Yüksek hacim ve hızda veri girişi yaratmak
  - Beklenenden çok daha yüksek frekansta kesmeler yaratmak
  - Beklenen ve kullanılandan çok daha fazla bellek ve işlemci gücü gerektirecek durumlar yaratmak
- Sistem donanımının bir kısmının çökmesi durumunda geri kalan bileşenlerin çalışmayı sürdürebilme durumlarının testi
- İşletim sistemini yükleyerek çökertmeye yönelik testler
- Sabit disk veya optik teker (CD) gibi yardımcı saklama ortamlarına aşırı miktarda veri erişimi yaptırarak yapılan testler
- Üzerinde işlem yapılan verilerin alacağı en düşük ve en yüksek sınır değerleri etrafında, algoritma sonuçlarının doğruluğunu gerçeklemek için yapılan testler
- Sistemin kaldırabileceği en fazla yük durumunda, ani etkilere verilecek tepki süresini ölçmek üzere yapılan testler.

Germe testleri, yükleme testlerinden biraz farklı olarak, isterlerde tanımlanmış değerlere göre, sistemin yüklendikten sonraki davranışlarını, güvenilirlik açısından değerlendirebilmek üzere yapılır.

### 12.2.4.3. Geri Kazanma Testi

Bilgisayar sistemlerinin çoğundan bir hata durumunda kendini toplayarak tekrar çalışmaya devam etmesi beklenir. Tabii ki, bir metin düzenleyici veya çizim yazılımı için kendiliğinden devamlılık mutlak bir ister değildir. Donanım hatası oluştuğunda, eğer yedekleme amaçlı, fazladan donanım bulunuyorsa, bu donanım otomatik olarak devreye girebilir. Bu tür donanımlar yüksek maliyetleri nedeniyle, uygulama alanındaki önem derecesine göre kullanılırlar. Bazı durumlarda, donanım yedeklemesi yanında, yazılım yedeklemesi yapılır veya hataya dayanıklı yazılım kullanılır. Yedekli

yazılım mimarisinde, ana yazılım birimi çalışırken, yardımcı yazılım da aynı veya farklı bir donanım üzerinde paralel şekilde çalışır, fakat çıktı üretmez; ana yazılımın çökmesi halinde de bu yardımcı yazılımın çıkışları otomatik olarak devreye girer. Hataya dayanıklı yazılım ise, herhangi bir nedenle bütünüyle çökmemek üzere, kendi kendini düzeltebilen modüller halinde geliştirildiğinden, kusurlu modül yeniden başlatılarak eski durumuna gelmesi sağlanır. Her iki yöntemde de geri kazanma (recovery) belirli bir süre içinde yapılmak zorundadır. Aksi takdirde, uygulama alanının gereksinimlerine göre, fazla bir tehlike olmayabilir veya ekonomik kayıplar, hatta can kayıpları meydana gelebilir.

Geri kazanma testi önce yazılımı, sonra da donanımı çeşitli olası şekillerde bilinçli bir şekilde çökerterek sistemin kendini tekrar toplamasının denenmesi, isterlerin doğrulanması amacıyla yapılır. Bu kapsamda genellikle şu testler yapılır:

- Yedekli donanım mimarisinde, asıl çalışan donanımın devreden çıkarılması halinde, yedek donanımın kesintisiz çalışmaya devam etmesi (hot-swapping)
- Yedekli yazılım mimarisinde, ana yazılımın devreden çıkarılması ve yardımcı yazılımın otomatik olarak devreye girmesi, bilgi işlemenin kayba uğramadığının kontrolü, arada geçen sürenin isterlere uygunluğu
- Yeniden yazılım modülü başlatma yönteminde, çöken modülün tekrar kalkındırılması ve çökmeden önce durumunu kazanması (veriler, işlevsel durum gibi)
- Çökme sırasında kaybolması olası verilerin tekrar alınması veya üretilmesi
- İnsan katkısı gereken geri kazanma durumlarında, ortalama onarım zamanının ölçülmesi, isterlere göre değerlendirilmesi (çökmüş yazılımın üzerinde çalıştığı bilgisayarı tekrar açıp kapamak veya yazılımı yeniden yüklemek gibi).

### 12.2.4.4. Emniyet Testi

Bazı bilgisayarlı sistemler uygulama alanının özelliklerine göre emniyetli olarak çalışmak zorundadırlar. Örneğin, önemli özkaynakları denetleyen sistemler, sağlık alanındaki sistemler, uçuş ve seyir sistemleri, askeri amaçlı sistemler geliştirilirken mutlaka emniyet ön planda tutulmalıdır. Bir yazılım kusuru nedeniyle, o anda arzu edilmeyen bir etkinliğin, bir işin yapılması çeşitli şekillerde engellenmelidir. Bu gibi durumlar önceden belirlenmeli ve önce test ortamında, sonra da gerçek ortamda, kontrollü bir şekilde test edilmelidir. Bu amaçla şu testler yapılabilir:

- Bir yazılım veya donanım kusuru oluştuğunda, sistemin yerine getirmesi gereken ana işlevde tehlike oluşturabilecek durum oluşup oluşmadığının testi
- Sonucunda bir füzenin ateşlenmesi gibi, o anda kesinlikle istenmeyen bir olayın meydana gelmesine yol açabilecek tetikleme zincirinin olup olmadığının testi
- Uç noktalardaki emniyet gereksinimleri için tasarım sırasında alınmış donanım veya yazılım düzeneklerinin testi (donanım kilidi, özel anahtar kullanımı, yazılımda şifre sorulması, onay alma, belirli durumlar oluşmadan bir işlevin gerçekleşmesine yönelik izni vermeme, merkezi bir kilit sistemi gibi).

### 12.2.4.5. Güvenlik Testi

Bazı bilgisayarlar hassas veya çok özel bilgileri işlerler. Örneğin, bir banka yazılımı binlerce müşterinin hesaplarını tutar. Bir sosyal güvenlik sistemi yazılımı yüzbinlerce, milyonlarca kişinin özel kayıtlarını tutar. Bir müzenin alarm sistemi çok değerli eşyaları hırsızlığa karşı korur. Bu sistemlerin koruduğu bilgiler veya nesneler, kanunsuz bir şekilde, yetkisiz kişilerce ulaşılıp kullanılmaya çalışılabilir. Bu amaçla bilgisayar saldırganları (hacker) veya virüsler çok büyük tehlike oluşturmaktadırlar. Sistem tasarımcıları, bir sistemin güvenliğini kırıp erişim sağlamak için kötü amaçlı kişilerin harcaması gereken maliyeti, korunan bilginin değerinden daha yüksek tutmak üzere yöntemler aramalı, bulmalı ve güvenli olarak uygulamalıdır. Bu tür tehlikelere açık olan sistemler için aşağıdaki güvenlik testleri yapılmalıdır:

- Ön kapı, yan kapı ve arka kapı denen sistem girişleri için tedbir alındığı, yetkisiz ve uygunsuz giriş çabalarının engellendiğinin ve rapor edildiğinin testi
- Şifrelerden başlayan ve her türlü giriş yöntemini deneyen bir testçi tarafından sistemin denenmesi
- Test yazılımları ile olası girişimler karşısında sistem davranışının incelenmesi
- Normal olmayan bir girişin fark edilmesi ve raporlanması testi
- Yetkisiz giriş denemelerini fark etme, girişimciyi bir tuzağa çekme, oyalama ve kimliğini tespit etmeye yönelik testler
- Uzman testçi kullanarak sistemin zayıf noktalarını kullanıma sunmadan önce bulmaya yönelik gerçek ortamda, fakat kontrollü olarak yapılan testler.

### 12.2.4.6. Başarım Testi

Her bilgisayar sistemi için en önemli genel isterlerden biri *başarım*, yani performanstır. Yazılımın yürütme anındaki başarımı, geliştirme sırasında uygulanan tekniklere ve tasarıma bağlı olduğu kadar üzerinde çalıştığı donanıma da bağlıdır. Ancak, bütün olarak geliştirilen sistemlerin en son başarımları özel olarak test edilmeli ve uygulama alanının özelliklerine göre uygun bir düzeyde tutulmasına çalışılmalıdır. Örneğin, büyük bir veritabanı yönetim sisteminin başarımı, arama ve sonucu gösterme işlevi için gereken süredir. Gömülü bir sistemin başarımı, insan katkısı olmadan yapmak zorunda olduğu işleri yapmasıdır. Özellikle gerçek zamanlı sistemler için tanımlanmış olan zaman kısıtlarına uymak hem yazılım hem de donanım bileşenleri tarafından karşılanması gereken çok önemli isterlerdir. Bu isterlerin uygun şekilde karşılanıp karşılanmadığını görebilmek için, tümleştirilen yazılım ve donanım üzerinde başarım testleri yapılır. Aslında başarım testi birim testleri sırasında başlar. Ancak, saydam kutu testlerinde başarılı olan bir birim, tümleştirildikten sonra farklı bir başarım gösterebilir. O nedenle, tüm sistemin genel başarımının testinin yapılması, isterlerin doğrulanması açısından çok önemlidir.

Başarım testleri bazen germe testleri ile beraber yapılır. Çünkü, başarım, sistem hafif yükteyken kabul edilebilir olsa bile, sistemin çokça yüklendiği durumlarda aşırı düşebilir. Böyle durumları belirlemek ve başarım derecesini ölçmek için çeşitli araç

ve gereçler kullanılır. Örneğin, bir verinin sisteme giriş anı ile çıkış anı arasındaki zaman farkının ölçülmesi, toplam bilgi işleme yeteneğini ölçmek üzere, bilgi işleme öğelerinin sayılması gibi testler bu gereçlerle yapılır. Bu şekilde, sistemin genel başarımı için bir ölçü değeri verilirken, mevcut özkaynaklar ile belirli girdilere karşı elde edilen çıktılar arasında sistemin yavaşlama fonksiyonunun elde edilmesine çalışılır. Bu fonksiyon ile, sistemin hangi yükte uygun başarımla çalıştığı belirlenir, daha fazla yükte nasıl yavaşlayacağı konusunda kullanıcı uyarılır.

11. Hafta
1. Grup
SON