

## 6. Yazılımın Test Edilmesi

Yazılımı bir sistem ve sinyaller organizasyonu olarak düşünmek gereklidir. Girdi sinyallerine karşı bir çıktı sinyalleri üretebilir; girdisi olmadan çıktı üretebilir; geri besleme ve girdiye bağlı olarak çıktı üretebilir.

Geliştirilen yazılımlarda, kaynağı ve sebebi değiştirmekle birlikte çeşitli hataların olması kaçınılmazdır. Çalışılan uygulama alanının kritikliğine göre bu hataların doğuracağı sonuçların biçimi değişebilmektedir. Bir finans uygulamasında yapılan küçük bir hata çok büyük miktarlarda para kaybına yol açabilecekken, bir askeri uygulamada yapılması muhtemel küçük bir hata mal kaybının yanı sıra can kaybına da neden olma riskini taşımaktadır. Bu nedenle **profesyonel kullanımı planlanan tüm yazılımların içerisindeki hataların bulunması ve düzeltilmesi gereklidir.**

*Yazılım test çalışmaları, geliştirilen ya da geliştirilmekte olan yazılımlar içerisindeki hataların bulunup, düzeltilmesi, yazılımların hata içermedinin garantilenmesi ve yazılımların doğru çalışığının gösterimi amacıyla gerçekleştirilen faaliyetlerdir.* Bu çalışmalar, geliştirme sürecinde gereksinimlerin belirlenmesi aşamasında başlayıp, yazılımların (Yazılım bir sistemi tanımlamamaktadır.) müşteriye teslim aşamasına kadar devam eden bir süreçte belirli disiplinlere uygun olarak yapılmasını gerektirmektedir.

Yazılım test faaliyetleri sadece geliştirilen kod parçaları üzerindeki hataların bulunması faaliyetlerini değil, geliştirme çalışmaları sırasında hata oluşmasını önleyecek yöntem ve yaklaşımların belirlenmesi faaliyetlerini de içerir. **Geliştirme sürecinin başlangıç aşamalarında tespit edilen bir hatanın maliyeti ile müşteriye teslim edilmiş bir sistem üzerinde tespit edilen bir hatanın maliyeti arasında ciddi farklar olabilmektedir.** Bu nedenle olası hataların geliştirme aşamasının mümkün olduğunda erken safhalarında bulunması için yapılan gözden geçirme ve tasarım doğrulama faaliyetleri gibi çalışmalar da doğrulama ve geçerlileme sürecinin bir parçasıdır.

Günümüz sistemlerinin tek bileşeni yazılım olmayan, yazılım dışında elektronik, mekanik ve elektromekaniksistemler, algılayıcılar, tartıcılar ve mukayes ediciler (filtreler) gibi bileşenleri de içerdiginden, çok sayıda uzmanlık grubu bir arada çalışmaktadır. Bu nedenle yazılımların doğrulama süreçleri içerisindeki aşamalar, çıktılar, roller ve kullanılan araçlar **hata tespit ve düzeltme faaliyetlerinin yanı sıra hata önleme amacıyla da dönük olmalıdır.**

Test, bir sistemi manuel veya otomatik yollarla deneyerek veya değerlendirerek, belirlenmiş gereksinimleri karşıladığından doğrulanması veya beklenen ile gözlenen sonuçlar arasındaki farkların belirlenmesi sürecidir.

*Yazılım testi ise bir yazılımın sonsuz sayıdaki çalışma alanından, sınırlı sayıda ve uygun şekilde seçilmiş testler ile beklenen davranışlarını karşılamaya yönelik, dinamik olarak yapılan doğrulama faaliyetlerini kapsamaktadır.* Bu kapsamda dikkat edilmesi gereken hususlar şunlardır:

- **Dinamik** (Dinamik: değişken; statik: sabit) olarak yazılım mutlaka çalıştırılarak test edilmelidir.
- Yazılımın neredeyse sonsuz sayıda olabilecek çalışma alanlarının tümünün testi imkânsız olacağından; kritiklik düzeylerine göre sıralanıp, **yeterli görülen sayıda, en kritikleri** test edilmelidir.
- Test edilecek davranışın doğasına uygun ve **muhtemel riskleri** göz önünde bulunduran testlerin gerçekleştirilmesidir.
- Test edilecek yazılımın, **kullanıcı bekentilerine**, gereksinimlerine ve akla uygun, mantıklı bekentilere cevap verebildiğinin test edilmesidir.

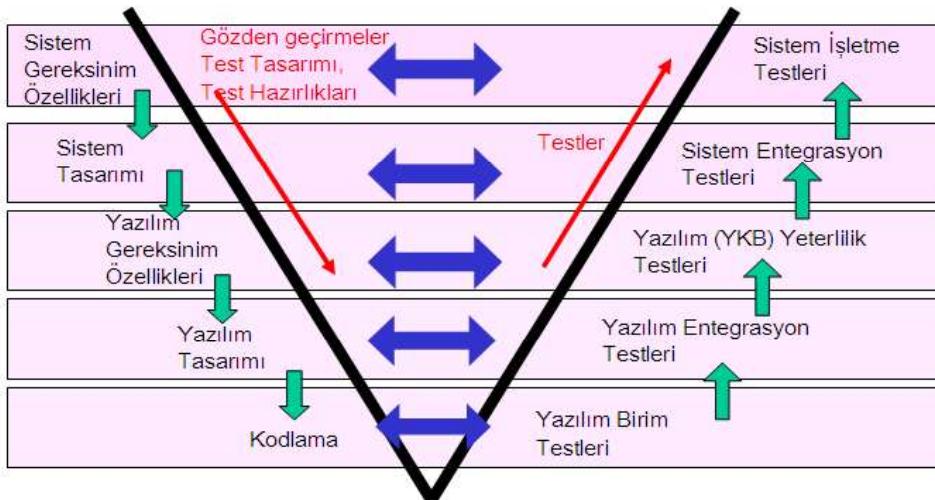
#### **İşlevsel testler:**

- uygunluk,
- tamlık,
- birlikte çalışma,
- hatalı girdi
- senaryo testlerini kapsar

Performans ve güvenilirlik testlerini kapsayan testler ise işlevsel olmayan testlerdir. Karşılaşılacak kullanıcı davranışına, veri hacmine uygun, gerçekçi bir şekilde **performans testleri** şekillendirilmelidir. **Güvenilirlik testleri ise olgunluk, hata-toleransi, toparlanma ile ilgili testleri kapsamalıdır.**

Test faaliyetlerinin, yazılım geliştirme sürecinin daha başlangıç safhalarından itibaren vazgeçilmez bir parçası olduğu açıklıktır. Bu nedenle yazılımın tasarıımı yapılırken, test planının da belirlenmiş olması, tasarımın test planına uygun özellikleri taşımاسının sağlanması gerekmektedir.

Yazılım test faaliyetleri, tüm yazılım geliştirme süreci boyunca devam eden, sadece hataların bulunup ayıklanması işlemlerini değil, hata oluşmasını önleyici yaklaşımların uygulanmasını da içeren faaliyetlerdir. Test faaliyetleri aşağıda popüler bir gösterim şekli olan V-Modeli ile gösterilmiştir. V modelinde her tasarım aşamasına karşılık bir test aşaması mevcuttur.



Yazılım Geliştirme sürecinde test ile şu adımlar tanımlanır:

- Yayınlanan Sistem Tasarım Tanımı ve Yazılım Gereksinim Özellikleri incelenerek, Test konusundaki genel yaklaşım ve test faaliyetlerinin amacı göz önünde bulundurularak test aktivitelerinin planı, ***Yazılım Test Planı***, hazırlanır.
- Yazılım test planı çerçevesinde test platformlarının (test düzeneklerinin), kontrolünü ve veri akışı bilgilerini, yazılım ve donanım test araçlarını (***simülatörler, test sürücüler***) ve veri hazırlıklarını içeren test alt yapısı gereksinimleri belirlenir. Test araçları sadece donanım olabildiği gibi donanım ve yazılımla birlikte de olabilir.
- Hazırlanan Yazılım Test Planı tanımlı sürece uygun olarak ***gözden geçirilir***. Yazılım mühendisinin kazanması gereken en önemli yeteneklerden biri gözden geçirme ve fark etmedir.
- ***Testlerin planlanması*** kapsamında yazılım ***testlerinin hazırlık aşamasında, sistem ortamın kurulmasında, ön testler sırasında, testlerin gerçekleştirilmeye aşamasında kimlerin sorumlu olacağı, testlerin gerçekleştirileceği ortamların planlanması nasıl yapılacağı, tanımlı süreç dışında*** projeye özel bir uygulama yapılmış yapılmayacağı, eğer farklılık varsa hangi aşamalarda ***süreçten farklılıklar*** olacağı, testler sonrasında ***raporlamanın nasıl ve kim tarafından gerçekleştirileceği, düzeltmelerin ve sonrasında testlerin nasıl yapılacağı*** gibi sorulara cevaplar verilir.

### **Testler:**

- ***İsteklerdeki eksiklikleri, hataları ve belirsizlikleri*** belirlemek
- Kişisel hatalar: Benciller, psikopatlar, kendine aşırı güvenenler, yalnız kurtlar.
- Geliştirilen Algoritma ve Matematiksel Modellerde eksiklikleri, hataları ve belirsizlikleri belirlemek. İlk iki aşamada problem %80 oranında çözümler.
- Yazılım yazılırken hatayı bulmak
- Yazılım geliştirme aşamasında donanımın bileşenleri test etmek
- Kod yazılım sonrası bütürleştirme aşamasında sistem performansını test etmek
- Yazılım bütürleştirme ve uygulama aşamasında yapılan testler
- Kod çalışmaya başladığı andan itibaren gözden kaçan isteklerin belirlenmesi
- Bakım ve onarım aşamasında testler
- Gerçek zamanlı çalışırken yapılması gereken testler

### **Testle İlgili Gözlemler:**

- “Test, hata bulma niyetiyle yürütülen bir program sürecidir.” - Myers
- “Test, hataların varlığını gösterebilir, ancak onların yokluğunu asla gösteremez.” - Dijkstra

### **İyi Test Uygulamaları:**

- İyi bir test uygulaması ne içerir? Simülasyon (Benzeri-model), Optimizasyon (Sınırlama, sapma bulma),
- İyi bir test durumu, programın doğru şekilde çalıştığını göstermeyen durumlar için bir hatayı, tespit edilmemiş bir kusuru belirleme olasılığı yüksek olan bir işlevdir.
- Mümkünse kendi programınızı test etmeyiniz, kendi hatanızı bulmada zorlanırsınız. Kendiniz test etmek zorunda iseniz, kod yazmaya başladığınız andan itibaren adım adım sonuç takibi yapan test uygulamaları gerçekleyin.
- Her test aşamasına beklenen sonucun elde edilip edilmediğinin tespit edilmesi gerekmektedir.
- Yeniden üretilemez veya sinek testinden kaçının. Öylesine bir kod yazılım gerçeklenmiştir ki, geri dönüş olduğunda yazanlarda ne yaptıklarını bilemez durumdalar, hatanın nerede olduğunu bulmak imkansız. Çünkü sistematik değil ve planlı değil. Kod parçalarına açıklamanın eklenmesi ve Readme (deneyim, fırsatlar ve hikaye içermeli) dokümanı oluşturulması gerekmektedir.
- Geçersiz ve geçerli giriş koşulları için test senaryoları yazınız.
- Her testin sonuçlarını iyice inceleyin
- Bir yazılım parçasındaki tespit edilen kusurların sayısı arttıkça, daha fazla tespit edilemeyen kusurların var olma olasılığı da artar.
- En iyi çalışanlarınızı test etmeye atayın
- Yazılım tasarımanızda test edilebilirliğin önemli bir amaç olduğundan emin olun.
- Programı daha kolay test etmek için asla değiştirmeyin
- Hemen hemen her diğer aktivite gibi testler hedeflerle başlamalıdır.

Yazılımlarda bellek sızıntısının ya da donanımsal hataların da olması durumu da sorun teşkil edebilecek önemli bir konudur. Birim testleri ile birlikte kod analiz araçları kullanılarak yazılımlar analiz edilmekte ve bellek sızıntıları tespit edilip düzeltilmektedir.

## 6.1. Yazılım Yeterlilik Testi

Yazılım yeterlilik testleri, kara kutu test (işlevlerin adım adım izlendiği durumların kaydı) yaklaşımıyla yazılımın, gereksinimlerini karşıladığıının doğrulanması amacıyla yapılır.

Yazılım Yeterlilik testleri, Yazılım Gereksinim Özellikleri dokümanı temelinde şekillenir ve planlamadan ardından hazırlanmaya başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Yayınlanan Yazılım Gereksinim Özellikleri temel alınarak ve testlerin gereksinimleri yeterli düzeyde kapsamasına özen gösterilerek test tanımları belirlenir.
- Testin başlatılması, yerine getirilmesi ve gerekli verilerin toplanması için gerekli olan adımların sıralamasını, her bir adım için beklenen sonuçları ve değerlendirme kriterlerini içeren test yönergeleri hazırlanır.
- Test tanımlarını ve yönergelerini içeren **Yazılım Test Tanımları** dokümanı tanımlı sürece uygun olarak gözden geçirilir. Bu gözden geçirme sürecinde kontrol listesi referans olarak kullanılabilir.

Yazılım yeterlilik test tanımlarının hazırlanması için öncelikle yazılım **gereksinimlerinin olgunlaşmış olması** gerekmektedir. **Gereksinimlerin tamamen veya kısmen belirsiz olması**, test sürecinin sağlam temeller üzerine oturmasını engeller. Bir gereksinimin değişmez hale getirilip dondurulmasının pratikte zor olduğu düşünüldüğünde uygulama için önerilebilecek çalışma yöntemi, yazılım gereksinimlerinin yazılımı geliştirenlerin deneyimleri doğrultusunda test edilebilecek olgunluğa erişmiş olduğu kanaatine varılması ile gereksinim dokümanının doldurulması, ve bir kopyasının alınıp, gereksinimlerin test edilebilecek düzeye erişip erişmediğinin kontrolü için test tasarım ekibinin görüşüne sunulmasıdır. Daha önceden aynı özelliklere sahip yazılımların testlerini muhtemelen gerçekleştirmiş olan test tasarım ekibi her bir gereğin nasıl test edileceğini ve test sırasında nasıl bir ortamın sağlanması gerektiği konusunda bütün gereksinim dokümanını gözden geçirmelidir. Test tasarım ekibi, gereksinimlerin test edilebilir olgunluğa erişmiş olduğunu onayladığında, doldurulmuş gereksinimlere göre test hazırlıklarına paralelden başlayabilir ve test hazırlıkları devam ederken bir yandan da gereksinimlerin olgunlaşması devam edebilir. Ancak test hazırlıklarının son aşamasında test tanımları ve gereksinimler arasında bir uyumsuzluk oluşup olmadığı kontrol edilmelidir.

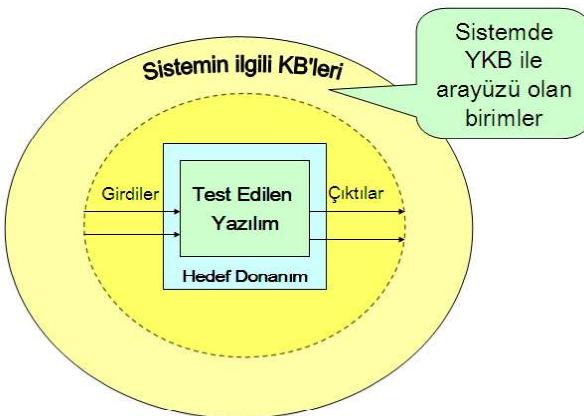
Yazılım gereksinimlerinin olgunlaşması ve test edilebilirliğe göre gözden geçirilmesi test tanımları hazırlanması aşamasına geçiş için olmazsa olmaz iki önkoşuldur: Test tanımları, doğruladığı gereksinimlerin yazılım tarafından karşılanıp karşılanmadığını bulmaya yönelik olmalı ayrıca olağan dışı girdileri de kapsayacak şekilde yazılmalıdır.

***Test tanımlarının yazılması sırasında dikkat edilmesi gereken bir diğer konu da her bir gereksinimin en az bir test tanımı ile doğrulanmasının gerekliliğidir.*** Gereksinim Yönetim araçları kullanılarak gereksinimler ve test tanımları arasında izlenebilirlik kurulup test edilmemiş herhangi bir gereksinimin kalmaması sağlanmaktadır.

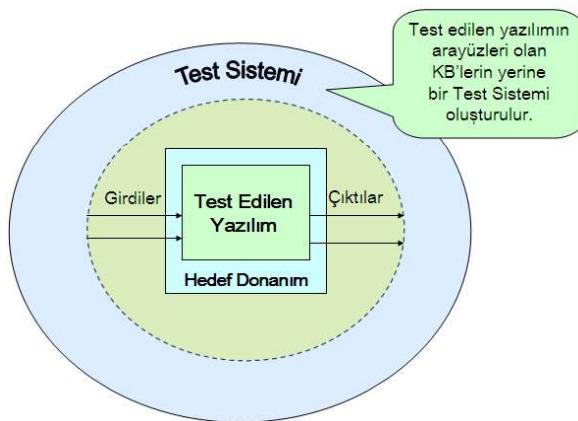
Hazırlanan Yazılım Test Tanımları dokümanında; testin amacı, testin hangi gerekleri adresleyerek doğruladığı, testin hangi konfigürasyonda gerçekleştirileceği, test önergesi, testin başarılı sayılması için gerekli olan kriterler ve varsa test girdileri, önkoşulları, sınırlamaları ile çıktıları tanımlanmaktadır. Test önergelerinin kapsamı ise doğrulanın gereğin kritikliği ve hata çıkma olasılığıyla doğru orantılı olmalıdır. Test sırasında çıkacak olası bir hatanın görevde etkisi ve sonuçları değerlendirilmelidir. Ayrıca testlerin belirlenmesinde; karmaşık, zaman içinde değişikliğe uğrayan, uygulama içinde yenilik olan (yöntem, yeni teknoloji vb.), optimizasyon gerektiren ve daha önce hata bulunan kısımlara özel önem verilmelidir.

## **6.2. Test Yazılımları Geliştirme Faaliyetleri**

Yazılım yeterlilik testleri için testlerin kara kutu ( İzleme, takip, kayıt) yaklaşımı ile gerçekleştirilebilmesi, yazılımın sistemden bağımsız bir test ortamı içerisinde koşması için önem taşımaktadır. Yazılımda oluşabilecek hataları, sistemde oluşabilecek hatalardan bağımsız olarak ayıklamak test işlemini kolaylaştırır, verimliliği artırarak, hatanın olabildiğince erken aşamalarda yakalanmasını sağlar. Bu durum test edilecek yazılımın beraber çalıştığı her türlü yazılımın ve donanımın gerçek sistem üzerinde çalışıormuşçasına simülle edilmesi ihtiyacını doğurmaktadır. Bunun için Şekil -A'de görülen test edilen yazılımın ilişki içinde olduğu konfigürasyon birimlerinin (KB) yerine Şekil-B'deki gibi test sistemi yerleştirilmektedir. Test sistemi test yazılımlarından oluşmaktadır. Test yazılımları test edilecek yazılım ile simülasyonunu yaptığı yazılım veya donanımın ilgili arayüzü kontrollü bir şekilde gerçekleyen yazılımlardır. Yani bir yazılımın testi sırasında o yazılımın haberleştiği yazılım veya donanımın simülasyonunun yapılması, o yazılım veya donanımın tüm işlevlerinin gerçekleştirilmesi değil sadece test için gerekli olan arayüzün kodlanmasıdır. Öte yandan test yazılımları, ilgili arayüzleri gerçeklerken, gerçek sistemde çalışan yazılımlardan farklı olarak o arayüzdeki olağan dışı durumları da gerçekleme olanağına sahip olmalıdır. Bir yazılımın tam anlamıyla test edilmesi için zaman zaman birden fazla ve farklı yazılım geliştirme ortamlarında kodlanmış test yazılımlarına ihtiyaç duyulabilmektedir.



Şekil-A: Yazılım Çalışma Ortamı



Şekil-B: Yazılım Test Ortamı

Gereksinimlerde yapılacak ufak bir değişiklik test yazılımlarında kökten değişikliklere yol açabilmektedir. Örneğin, test edilen yazılımın gereksinimlerinde olabilecek bir değişiklik test yazılımının tamamen başka bir ortamda tekrar geliştirilmesi gerektiğini ortaya çıkarabilmektedir. Bu da işgücü ve zaman kaybına yol açabilmektedir. Bu yüzden testlerin planlama aşamasından sonra tasarım ekipleri tarafından gözden geçirilmesi önem kazanmaktadır.

Test yazılımlarının hazırlanması aşamasının en önemli girdileri yazılım test tanım dokümanı ve yazılım arayüz tasarım tanımı dokümanlarıdır. Her iki dokümanın da olgunlaşması test yazılımlarının geliştirilmesinin başlaması için gereklidir.

Otomatik test yazılımlarının kullanılması ile testlerin daha hızlı gerçekleştirilmesi sağlanabilmektedir. Bu amaçla kaydet-oynat yeteneğine sahip, kullanıcının, kullanıcı arayüzünde yaptığı davranışları kaydederek daha sonra sanki kullanıcı yapıyormuş gibi aynı hareketleri farklı girdilerle defalarca tekrarlayabilen araçlar kullanılabilmektedir. Ancak farklı

arayüzler üzerinden farklı ortamlara sahip çevre birimlerle etkileşimi çok olan gömülü yazılımların testi için bu gibi otomatik test araçlarının yetenekleri kısıtlı kalmaktadır. Bu gibi durumlarda pratik olarak önerilen yöntem, geliştirilmiş olan test yazılımı içerisinde bu otomasyonun gömülmesi ve test yazılımı aracılığı ile testlerin mümkün olduğunda otomatik olarak gerçekleştirilmesinin sağlanmasıdır.

### 6.3. Testlerin Gerçekleştirilmesi

Kodun hazır olduğunu duyurulmasını takiben Yeterlilik testleri gerçekleştirilmeye başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Test tanımları ve test araçları olgunlaştırılır.
- Yazılım gereksinimlerinden test tanımlarına izlenebilirlik kontrol edilir.
- Üzerinde koşacağı Donanım Konfigürasyon Birimi hedef prototipi ve test yazılım ve araçlarının bir araya getirilmesiyle bir test düzeneği kurulur.
- Test tanımları çerçevesinde yeterlilik testleri gerçekleştirilir. Hata oluşması durumunda problemlerin çözülmesi amacıyla ilgili sürecة geçilir ve süreç işletilir.

Testler gerçekleştirilmeye başlanmadan önce ortamın (test konfigürasyonunun) testlere hazır olup olmadığından kontrolünün yapılması gereklidir. Hazırlanan test yazılımlarının testler öncesinde doğrulanmış olması önem taşımaktadır. Resmi testler öncesi test edilecek yazılımin test ortamına entegrasyonun gerçekleştirildiği ve yazılımin testlere hazır olup olmadığından kontrolü duman (smoke) testleri aracılığıyla yapılır. Bu testler, test edilecek yazılımin en temel yeteneklerini doğrulayan test tanımları arasından seçilmelidir. Ancak bu testler başarılı olarak gerçekleştirirse, resmi olarak testlere başlanmalıdır.

Testlerin hangi sırayla yapılacağı doğrulanın testlerin kritikliği ve testlerin fonksiyonel olarak birbirleri ile olan bağlantılarıyla alakalı olabilir. Bu sıranın ne olacağına test planlama aşamasında karar verilmeli ve onaylanmalıdır.

Yazılıma yapılan ekleme veya düzeltmeler yeni hatalara sebep olabilmektedir. Bu hataların tespit edilmesi ve olası gerilemenin belirlenmesi amacıyla yapılan testler regresyon (gerileme) testleridir. Herhangi bir değişiklik sonucunda, yeni sürümü yapılan yazılımin **regresyon testlerinde**, sadece yeni sürümdeki değişiklik bilgisinin incelemesi sonucunda gerekli olduğunu belirlenen testler yapılabilir.

## **6.4. Test Seviyeleri**

- 1) Birim Testi
- 2) Entegrasyon - Bütünleştirme Testi
- 3) Doğrulama Testi
  - a. Gerileme testi
  - b. Alfa Testi
  - c. Beta testi
- 4) Kabul testleri

### **Birim Testi**

- Algoritmalar ve mantık
- Veri yapıları (global ve yerel)
- Arayüzler
- Bağımsız yollar
- Sınır şartları
- Hata işleme

### **Birim Testi, Birim Entegrasyon Testi**

Geleneksel yaklaşımada daha çok görsel denetimle ya da manuel yöntemlerle yapılan birim testleri, otomatik test araçlarının kullanımıyla daha sağlıklı ve tekrarlanabilir olarak yapılmaya başlanmıştır. Test planının ve test tanımlarının hazırlanmasına paralel olarak, yazılımların kodlanması ile birlikte birim testleri gerçekleştirilmeye başlanmaktadır. Yazılım Geliştirme Sürecinde Birim Testlerine yönelik olarak süreçte şu adımlar tanımlanmıştır:

- Yazılım Birim test hazırlıkları yapılır. Birim, modül ve bunların çeşitli bileşenlerinden oluşan kümelerin hangi sırayla test edileceği, test gereksinimleri (donanım ve yazılım), durumları ve yöntemleri belirlenir.
- Hazırlık kapsamında eğer gerekiyor ise sürücü/kart entegrasyonu ve bunların testi gerçekleştirilir. Kart seviyesinde koşan yazılımların ve geliştirilen veya hazır alınan kartın spesifikasyonlarına uygunluğu doğrulanır.
- Üretilen kod için birim/modül/küme testleri uygun debug/test araçlarıyla gerçekleştirilir. Testlerin kayıtları tutulur.
- Yazılım Konfigürasyon Birimi yeterlilik testi için yazılımın hazır olduğuna karar verilmesi durumunda kodun hazır olduğu duyurulur.

Birim testleri bu süreç dahilinde manuel ya da test araçları kullanılarak otomatik olarak yapılmaktadır. Test araçları kullanılarak otomatik test yapılması hedeflendiğinde, kullanılacak olan aracın kodlama aşamasına geçilmeden önce belirlenmiş olması ve aracın özelliklerine uygun şekilde yazılımın gerçekleştiriminin yapılması gerekmektedir. Otomatik test araçları yazılım birimleri için yazılan birim testlerinin işaretlenmesi ve bu testlerin

dışarıdan erişilerek çalıştırılması mantığı ile çalışmaktadır. Bu araçlar test yordamlarının işaretlenmesini sağlayan işlevleri ve doğrulama cümlelerinin yazılabilmesini sağlayan (karşılaştırma işlemleri vs.) yordamları içeren kütüphaneyi ve testlerin gerçekleştirimini ve raporlanmasılığını sağlayan kullanıcı arayüzüni içermektedir.

Birim testi için manuel yöntemler kullanılması durumunda yazılım geliştirme ortamının **hata ayıklama (debug) yeteneklerinden** faydalанılmaktadır. Performansa yönelik testler için ise koda eklemeler yapılarak zaman bilgisi tutulmakta, yazılım işlevini tamamladıktan sonra bu bilgiler diske kaydedilerek değerlendirilmesi manuel olarak yapılmaktadır.

## Sistem Entegrasyon Testi

Yazılım yeterlilik testlerinin gerçekleştirilmesi sonrasında sistem tarafından da çalışmalar Sistem Entegrasyon Testi Hazırlıklarını Yap aşaması ile devam eder. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Her bir entegrasyon aşamasındaki gereksinimlere uygun olarak test tanımları Test mühendisleri tarafından belirlenir ve gerekli test araçları oluşturulur. Tasarlanması gereken özel amaçlı donanım ve yazılım test araçları ve yeni test yatırımları için Donanım Geliştirme Sürecine uygun olarak planlama yapılır.
- Test tanımlarının dokümanente edildiği dokümanlar tanımlı sürece uygun olarak gözden geçirilir.
- Temin edilen birimler bütünselik sistem elde edilene kadar ihtiyaca göre aşamalı olarak entegre ve test edilir.

Test hazırlıklarının tamamlanması sonrasında Sistem/Alt sistem Testini Yap ve Entegrasyon Test Sonuçlarını Gözden Geçir aşaması gerçekleştirilir. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Entegre edilen her test edilecek yapı üzerinde hazırlanan dokümanlara uygun doğrulama testleri uygulanır ve test sonuçları kaydedilir. Bu kayıtların analizi ve yorumlanması ile test raporu hazırlanır.
- Hazırlanan rapor tanımlı sürece uygun olarak gözden geçirilir.
- Sistem tasarımı sürecinde belirlenen ve Sistem / Alt Sistem Tasarım Tanımı (Sistem Tasarım Tanımı) dokümanında temel özellikleri oluşturulan DONANIM KONFIGÜRASYON BİRİMİ ve YAZILIM KONFIGÜRASYON BİRİMİ'ler geliştirildikten veya hazır olarak temin edildikten sonra DONANIM KONFIGÜRASYON BİRİMİ ve YAZILIM KONFIGÜRASYON BİRİMİ'lerin entegrasyonu aşamasına geçilir. Entegrasyonun doğrulanması amacıyla entegrasyon testleri gerçekleştirilir.

Entegrasyon testleri kapsamında öncelikle donanım ve yazılım arayüzlerinin doğruluğu kontrol edilir. Arayüzler doğrulandıktan sonra SISTEM TASARIM TANIMI dokümanında belirtilen sistem çalışma senaryoları kapsamında fonksiyonel ve performans testleri yapılır. Test sonuçları Sistem / Altistem Test Raporu (STER) ile raporlanır. Test sonuçlarına göre DONANIM KONFIGÜRASYON BIRIMI ve YAZILIM KONFIGÜRASYON BIRIMI'ler için gerekli düzeltmeler yapılır.

### Neden entegrasyon testi gereklidir?

- Bir modül diğerini olumsuz etkileyebilir.
- Bütünleştirme yapıldığında, alt fonksiyonlar istenen ana fonksiyonu üretmemeyebilir
- Bireysel olarak kabul edilebilir hesaplamardaki uygunsuzluk kabul edilemez seviyelere yükseltilebilir.
- Birim testinde tespit edilmeyen arayüz hataları görünebilir
- Zamanlama problemleri (gerçek zamanlı sistemlerde) birim testiyle tespit edilemez
- Kaynak uyumsuzluk sorunları birim testi ile tespit edilemez

### Yukarıdan Aşağıya Entegrasyon:

1. Ana kontrol modülü bir sürücü olarak kullanılır ve ana modüle doğrudan bağlı tüm modüller için taslaklar kullanılır.
  2. Seçilen entegrasyon yaklaşımına bağlı olarak (önce derinlik veya genişlik), alt diziler birer birer birer birer değiştirilir.
  3. Her bir modül entegre edildiğinden testler çalıştırılır.
  4. Bir dizi testin başarılı bir şekilde tamamlanması üzerine, başka bir birim modülle değiştirilir.
  5. Yeni modüllerin entegrasyonu sonucunda hataların oluşmadığından emin olmak için regresyon testi yapılır.
- 
- Developing stubs that can pass data up is almost as much work as developing the actual module

### Yukarıdan aşağıya entegrasyon ile ilgili sorunlar

- Çoğu zaman, hesaplamalar hiyerarşinin altındaki modüllerde yapılır.
- Alt modüller tipik olarak daha yüksek modüle veri iletmez
- Alt seviye modüllere kadar geçikme testleri bir anda ziyade aynı anda pek çok modülün bütünlük sonuçları hazırlanır.
- Verileri aktarabilen taslakların geliştirilmesi, neredeyse gerçek modülün geliştirilmesi kadar bir iştir.

### **Alttan Entegrasyon**

- Entegrasyon, belirli bir yazılım alt işleyişini gerçekleştiren kümeler halinde birleştirilen ya da inşa edilen en alt düzey modüller ile başlar.
- Giriş ve çıkış test durumlarını koordine etmek için Sürücü yazılımları (kılavuzlar olarak geliştirilen kontrol programları) geliştirilir.
- Küme sınamıştır
- Sürücüler kaldırılır ve kümeler, program yapısında yukarı doğru birleştirilir

### **Alt Yukarı Entegrasyon Problemleri**

- Son modül bütünüleşene kadar tüm program mevcut değildir.
- Süreç ve kaynak çekişme sorunları süreçte geç kalana kadar bulunmaz.

### **Doğrulama Testi**

- Yazılımin YAZILIM İSTEKLERİ TANIMLAMA'de tanımlanan tüm gereksinimleri karşılayıp karşılamadığını belirler
- Yazılı gereklilikleri şarttır
- Yazılımın, yazılımdaki değişiklikler ve değişiklikler işliğinde tüm gereksinimlerini karşılayıp karşılamadığını belirlemek için regresyon testi yapılır.
- Regresyon testi, yeni testler geliştirilmeden, varolan validasyon testlerini seçici olarak tekrarlamayı içerir.

### **Alfa ve Beta Testi:**

- Müşterilere, odaklanmalarını istediğiniz şeylerin taslağını ve yürütütmeleri için belirli test senaryoları sunmak en iyisidir.
- Keşfettikleri kusurları düzeltme taahhüdüyle aktif olarak ilgilenen müşterilere sağlar.

## **6.5. Sistem Kabul Testi ve Arazi Testleri**

Tasarım ve doğrulama süreci tamamlanan sistemin müşteri isteklerini karşılayıp karşılamadığı kabul testleri ile ispat edilir. Müşteri tarafından belirlenmiş bütün gereklerin doğrulanması için Muayene Kabul Dokümanı oluşturulur.

Sistem bu dokümana göre önce şirket içerisinde doğrulandıktan sonra müşteri davet edilir ve testler müşteri ile birlikte tekrarlanır.

Sistem kabul testlerinin, sistem özelliklerine bağlı olarak şirket içerisinde ve/veya arazide yapılması gerekebilir. Örneğin bir silah sisteminde sistemin en önemli özelliği olan atış yeteneğinin gerçek atışlar yapılarak doğrulanması gereklidir. Bu gibi durumlarda testler için uygun atış alanlarının kullanılması gerekmektedir.

### **Kabul testleri**

- Müşterilerin mevcut veya doğrudan dahil olması haricinde, doğrulama testine benzer.
- Genellikle testler müşteri tarafından geliştirilir

## **Sistem Yeterlilik Testi**

Sistem entegrasyon ve testleri tamamlandığında Sistem İşletme Testlerinin yapılması ile sistemin doğrulanması sağlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Sistem İşletme Test Planı içinde yer alan test gereksinimlerinin sağlanmasında kullanılacak olan test tanımları yazılır.
- Sistem, gerçek kullanım ortamında veya benzer koşullarda hazırlanan test tanımlarına uygun olarak test edilir ve rapor oluşturulur.
- Raporlar Gözden Geçirme Sürecine göre gözden geçirilir ve doğrulanmış ve geçerli kılınmış sistem elde edilir.

Sistem yeterlilik testlerine yönelik çalışmalar Sistem Gereksinim Özellikleri dokümanı hazırlanıktan ve gözden geçirme süreci tamamlandıktan sonra başlar.

Sistem İşletme Test Tanımı dokümanına her bir gereksinim için doğrulama metoduna uygun olarak test tanımları yazılır ve testler uygulanır.

Yazılım tasarımlı yoğun olan sistemlerde Donanım Konfigürasyon Birimi'lerin sistemde doğrulanması entegrasyon testleri sonrasında tamamlanmasına rağmen yazılım ile ilgili gereksinimlerin doğrulanması sistem işletme testleri sonunda tamamlanmaktadır.

## 6.6. Hata Önleyici Faaliyetler

Doğrulama faaliyetlerinin önemli bir parçası hatayı olmadan önlemeye yönelik çalışmaların yürütülmesidir. Bu çalışmalar, ileride ortaya çıkabilecek hataları önleyerek, test ve düzeltme maliyetlerini azaltmayı hedeflemektedir. Bu kapsamında yapılan başlıca faaliyetler gözden geçirme ve tasarım doğrulaması faaliyetleridir.

Gözden geçirme, sürecin her safhasında farklı ürünler için uygulanabilen bir yöntemdir. Yazılım geliştirme sürecinde hazırlanan dokümanlar, yazılım mimari tasarımları, detaylı tasarımlar ve yazılan kodlar için gözden geçirme gerçekleştirilmektedir.

Gözden geçirme çalışmaları için süreçte tanımlanan temel adımlar şunlardır:

- Ürünün gözden geçirilmeye çıkarılması: Gözden geçirmeye hazır olduğu düşünülen ürün üzerinde değişiklikler dondurularak, gözden geçirme için belirlenen ekibe duyurulur. Belirlenen süre içerisinde ürünün incelenmesi ve gözden geçirme kaydının doldurulması istenir. Bu kayıtlar doldurulurken ürün için hazırlanmış olan kontrol listeleri de referans alınır.
- Gözden geçirme kayıtlarının incelenmesi: Belirtilen süre sonunda tüm gözden geçirme kayıtları ürünü hazırlayan kişi tarafından toparlanır ve kayıtların görüşülmesi için toplantı düzenlenir. Toplantı sırasında kayıtlar üzerinden geçilerek ürüne yansıtılacak değişiklikler belirlenir.
- Gözden geçirme kayıtlarının işlenmesi: Gözden geçirme toplantısı sırasında alınan kararlar doğrultusunda üründe değişiklikler yapılır.

Bu çalışmalar, ürünlere farklı paydaşların gözüyle bakarak erken safhalarda hataya sebep olabilecek hususların belirlenmesini ve düzeltilmesini sağlar.

Hatanın önlenmesine yönelik diğer faaliyetler tasarım doğrulama faaliyetleridir. Bu faaliyetler kapsamında, tasarım yapıldıktan sonra sistem senaryoları ele alınarak tasarımın tüm senaryoları karşılayıp karşılamadığı değerlendirilmektedir. Kullanıcı arayüzü tasarımlarının doğrulanması için ise hızlı prototipleme çalışması yapılarak ekranlar oluşturulmaktadır. Bu ekranlar üzerinde, yine senaryolar çalıştırılarak, işlevlerin akış sırasının ve ekranların amacına uygunluğu gözden geçirilir.

Bu yüzden testlerin planlama aşamasından, hazırlanma, gerçekleştirmeye ve raporlanma aşamalarına kadar geçen sürede uzman kişiler rol almaktır, basit ve anlaşılır iş talimatları ile test süreci desteklenmektedir. Testler yapılırken mümkün olduğunda hazır veya projeye özgü geliştirilen test yazılımlarından destek alınmaktadır. Hatalar, hata takip araçlarından takip edilmektedir. Test tanımları gereksinim yönetim araçlarına girilerek gereksinimlerle izlenebilirlikleri sağlanıp test edilmemiş gereksinimlerin kalmaması garanti edilmektedir. Tüm bu doğrulama faaliyetleri sistemlerin güvenilir bir şekilde müşteriye ulaşmasını sağlamaktadır.

## **6.7. Test Metotları**

- Beyaz kutu veya cam kutu testi
- Kara kutu testi
- Artımlı entegrasyon gerçekleştirmek için yukarıdan aşağıya
- Bir müşteri gibi davranışma

## **6.8. Test Tipleri**

- Fonksiyonel testler
- Algoritmik testler
- Pozitif testler
- Olumsuz testler
- Kullanılabilirlik testleri
- Sınır testleri
- Başlangıç / Kapama testleri
- Platform testleri
- Yük / stres testleri

### **Eşzamanlı Geliştirme / Doğrulama Test Modeli**

- Gelişme devam ederken gayri resmi doğrulama yapmak
- Yazılım geliştirme sürecinde, doğrulama testlerinin geliştirilmesi ve hata ayıklanması için bir fırsat sunar.
- Yazılım mühendislerine erken geri bildirim sağlar
- Resmi validasyondaki sonuçlar daha az olağandır, çünkü problemlerin çoğu zaten tespit edilmiş ve tespit edilmiştir.

### **Doğrulama Hazırlık Değerlendirmesi**

- Resmi olmayan doğrulama sırasında geliştiriciler, Yazılım İstekleri Tanımlama'ya uymak için gerekli değişiklikleri yapabilirler.
- Resmi olmayan doğrulama sırasında testleri Yazılım İstekleri Tanımlama'ya uygun hale getirmek için testleri çalıştırır ve gerekli değişiklikleri yapar.
- Resmi doğrulama sırasında yapılabilecek tek değişiklik, resmi doğrulama testi sırasında bildirilen hatalara yanıt olarak hata düzeltmeleridir. Şu anda yeni özellikler eklenemez.
- Resmi doğrulama sırasında, informal doğrulama sırasında aynı test seti tekrar çalıştırılır. Yeni test eklenmez.

### **Resmi Doğrulama Testi için Giriş Kriterleri**

- Yazılım geliştirme tamamlandı (“tamamlandı”)ının kesin tanımı gereklidir.
- Test planı gözden geçirildi, onaylandı ve belge kontrolü altında.
- Yazılım İstekleri Tanımlama'da bir gereksinim denetimi gerçekleştirılmıştır.
- Yazılım Tasarım Tanımları'erde tasarım denetimleri gerçekleştirılmıştır (Yazılım Tasarım Tanımları).
- Tüm “kritik modüller” üzerinde kod denetimleri gerçekleştirılmıştır.
- Tüm test komutları tamamlandı ve yazılım doğrulama testi prosedürü dokümanı gözden geçirildi, onaylandı ve belge kontrolü altına alındı.
- Seçilen test komutları gözden geçirildi, onaylandı ve belge kontrolü altına alındı.
- Tüm test senaryoları en az bir defa gerçekleştirılmıştır.
- CM araçları yerinde ve tüm kaynak kodu yapılandırma kontrolü altında.
- Yazılım problemi raporlama prosedürleri yürürlüktedir.
- Doğrulama testi tamamlama kriterleri geliştirilmiş, gözden geçirilmiş ve onaylanmıştır.

### **Resmi Doğrulama**

- Resmi olmayan doğrulama sırasında yürütülen aynı testler tekrar yürütülür ve sonuçlar kaydedilir.
- Yazılım Sorunu Raporları (Yazılım Problem Raporlama'ları) başarısız olan her bir test için gönderilir.
- Yazılım Problem Raporlama izleme gerçekleştirilir ve tüm Yazılım Problem Raporlama'ların durumunu içerir (ör. Açık, sabit, doğrulanmış, ertelenmiş, bir hata değil)
- Düzeltilen her bir hata için, Yazılım Problem Raporlama, hatayı düzeltmek için değiştirilen modülleri tanımlar.
- Temel değişiklik değerlendirmesi sadece değişmiş olması gereken modüller ve yeni özelliklerin girmediğinden emin olmak için kullanılır.
- Yeni hataların getirilmediğinden emin olmak için, informal kod incelemeleri değiştirilen modüller üzerinde seçici olarak yürütülmektedir.
- Hata bulup düzeltmek için gereken süre (find-fix cycle time) takip edilir.

Regresyon testi aşağıdaki yönergeler kullanılarak gerçekleştirilir:

- Hangi modüllerin ek testlere ihtiyaç duyabileceğini belirlemek için karmaşıklık önlemlerini kullanın
- Hangi testlerin tekrar çalıştırılacağına karar vermek için karar verin
- Yazılım tasarımı ve geçmiş tarih bilgisine dair temel karar
- Test durumunu izle (örn. Geçti, başarısız oldu veya çalışmadi).
- Yazılım güvenilirliği büyümeye izlemesi için kümülatif test süresini (gerçek testin toplam saati) kaydedin.

### **Doğrulama Testi için Çıkış Ölçütleri**

- Tüm test senaryoları yürütüldü.
- Tüm Yazılım Problem Raporlama'lar tatmin edici bir şekilde çözülmüştür. (Çözünürlük, hataların düzeltilmesi, daha sonraki bir yayına ertelenmiş, hata olmamasına karar verilmiş vb. İçerebilir). Tüm taraflar bu kararı kabul etmelidir. Bu kriter, tüm yüksek öncelikli hataların sabitlenmesi gerektiğini belirtirken, düşük öncelikli hataların durum bazında ele alınabileceğini belirtmek için tanımlanabilir.
- Yazılım Problem Raporlama'lar sonucunda yapılan tüm değişiklikler test edilmiştir.
- Yazılımla ilişkili tüm belgeler (Sistem Gereksinim Tanımlama, Yazılım Tasarım Tanımları, test belgeleri gibi) doğrulama testi sırasında yapılan değişiklikleri yansıtacak şekilde güncellendi.
- Test raporu gözden geçirildi ve onaylandı.

## **6.9. Test Planlama**

### **Test Planlama**

- Test Planı - Yapılacak işin kapsamını tanımlar
- Test Prosedürü - yürütülecek tüm bireysel testleri (test komut dosyaları) tutan bir kapsayıcı belgedir
- Test Raporu - test komut dosyaları çalıştırıldığında neler olduğunu belgeler
- Cevaplanacak sorular:
  - Kaç test gerekli?
  - Bu testleri geliştirmek ne kadar sürer?
  - Bu testleri yapmak ne kadar sürer?
- Ele alınacak konular:
  - Test tahmini
  - Test geliştirme ve resmi doğrulama
  - Doğrulama hazırlık incelemesi ve resmi doğrulama
  - Test tamamlama kriterleri

### **Test Tahmini**

- Gerekli test vakalarının sayısı şu şekildedir:
  - Yazılım Gereksinim Tanımlamadaki tüm fonksiyonların ve özelliklerin test edilmesi
  - Aşağıdakiler dahil olmak üzere uygun sayıda Bir Müşteri Gibi testi:
    - Yanlış yap
    - Yanlış veya hatalı giriş kombinasyonları kullanın
    - Yeterince yapma
    - Hiçbir şey yapma
    - Çok fazla yap
  - Bazı test kapsamı hedefine ulaşması
  - Yazılım güvenilirliği hedefine ulaşması

### **Test Tahmininde Dikkat Edilecek Noktalar**

- Test Kompleksliği - Birkaç büyük Sistem Tasarım Tanımı daha küçük olanlara sahip olmak daha iyidir.
- Farklı Platformlar - Farklı platformlar, işletim sistemleri vb. İçin testlerin değiştirilmesi gerekiyor mu?
- Otomatik veya Manuel Testler - Otomatik testler geliştirilecek mi? Otomatik testler oluşturmak için daha fazla zaman alır, ancak çalıştmak için insan müdahalesi gerektirmez.

### **Test prosedürü**

- Test senaryolarının toplanması
- Her test komut dosyasının ayrılmaz bir parçası beklenen sonuçtur
- Test Prosedürü dokümanı, testlerin daha kolay tekrarlanabilmesi için her testin kesintisiz ve temiz bir kopyasını içermelidir.

## **6.10. Testlerin Sonuçlarının Raporlanması**

Test sonuçları Yazılım Test Raporları ile raporlanmakta ve ilgili kişilere duyurulmaktadır. Testler sırasında tespit edilen hatalar, Hata Takip aracına girilmekte ve problemin ilgili kişiye atanmasından doğrulanmasına kadar bu araç üstünden takibi gerçekleştirilmektedir. Hataların kayıt altına alındığı Hata Takip araçları raporlama konusunda birçok olanak sunmaktadır. Hata Takip aracına girilen bir yazılım hatası, yazılım sorumlusu tarafından incelenmekte, yazılımda gerekli düzeltme yapıldıktan sonra girilen hatanın düzeltildiği aynı araç yoluyla bildirilmektedir. Yine aynı araç üzerinden hatanın düzeltildiği bilgisini alan test sorumlusu hatanın gerçekten düzeltildiğini yazılımın yeni sürümü üzerinde ilgili testi tekrar gerçekleştirerek hatanın giderilip giderilmediğini doğrulamaktadır. Özette, test sorumlusu ile yazılım sorumlusu arasındaki hata takibi, Hata Takip aracı kullanılarak yapılmaktadır. Böylelikle hataların sağlıklı bir şekilde kayıt altına alınması sağlanmaktadır.

Yeterlilik testlerinin tamamlanmasının ardından test kayıtları kullanılarak Yazılım Test Raporu hazırlanır. Test raporları Yazılım Konfigürasyon Birimi sorumluları ve kalite temsilcileri tarafından incelenerek Yazılım Konfigürasyon Birimi yeterlilik testinin tamamen yapılmış yapılmadığı, Yazılım Konfigürasyon Birimi'nin yeterliliği ve Sistem/Alt Sistem Entegrasyon ve Test aşamasına hazır olup olmadığı değerlendirilir.

### **Test raporu**

- Her bir test komut dosyasının tamamlanmış kopyası, icra edildiği kanıtlanır (yani testi yapan kişinin imzasıyla)
- Her Yazılım Problem Raporlama çözüldüğünü gösteren kopyasıdır.
- Açık veya çözümlenmemiş Yazılım Problem Raporlama listesi
- Her taban çizgisinde bulunan Yazılım Problem Raporlama'ların tanımlanması, her bir taban çizgisinde toplam yazılım problem raporlama sayısı
- Her bir yazılım taban çizgisi için yürütülen regresyon testleri

### **Doğrulama Test Planı, IEEE – Standard 1012-1998**

1. Genel bakış
  - a. organizasyon
  - b. Görevler ve Takvimler
  - c. Sorumluluklar
  - d. Araçlar, Teknikler, Yöntemler
2. Süreçler
  - a. Yönetim
  - b. Edinme
  - c. Arz
  - d. Gelişme

- e. Operasyon
  - f. Bakım
3. Raporlama Gereksinimleri
  4. İdari Gereksinimler
  5. Belge Gereksinimleri
  6. Kaynak Gereksinimleri
  7. Tamamlama Kriterleri