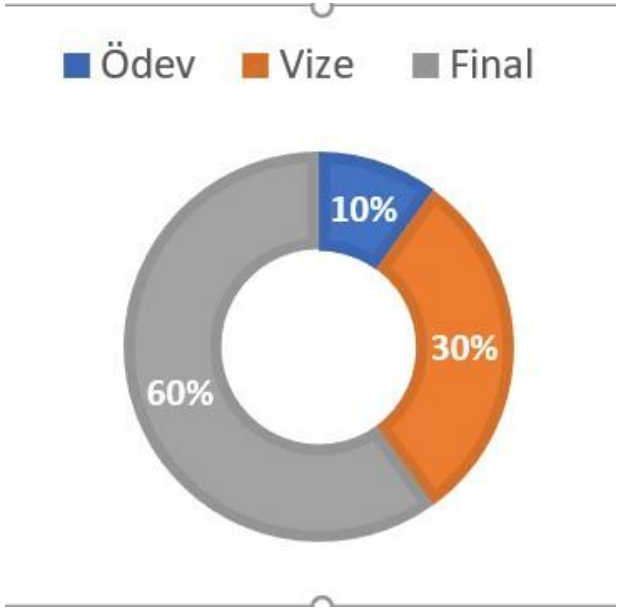
Yazılım Mühendisliği
1906003082015

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

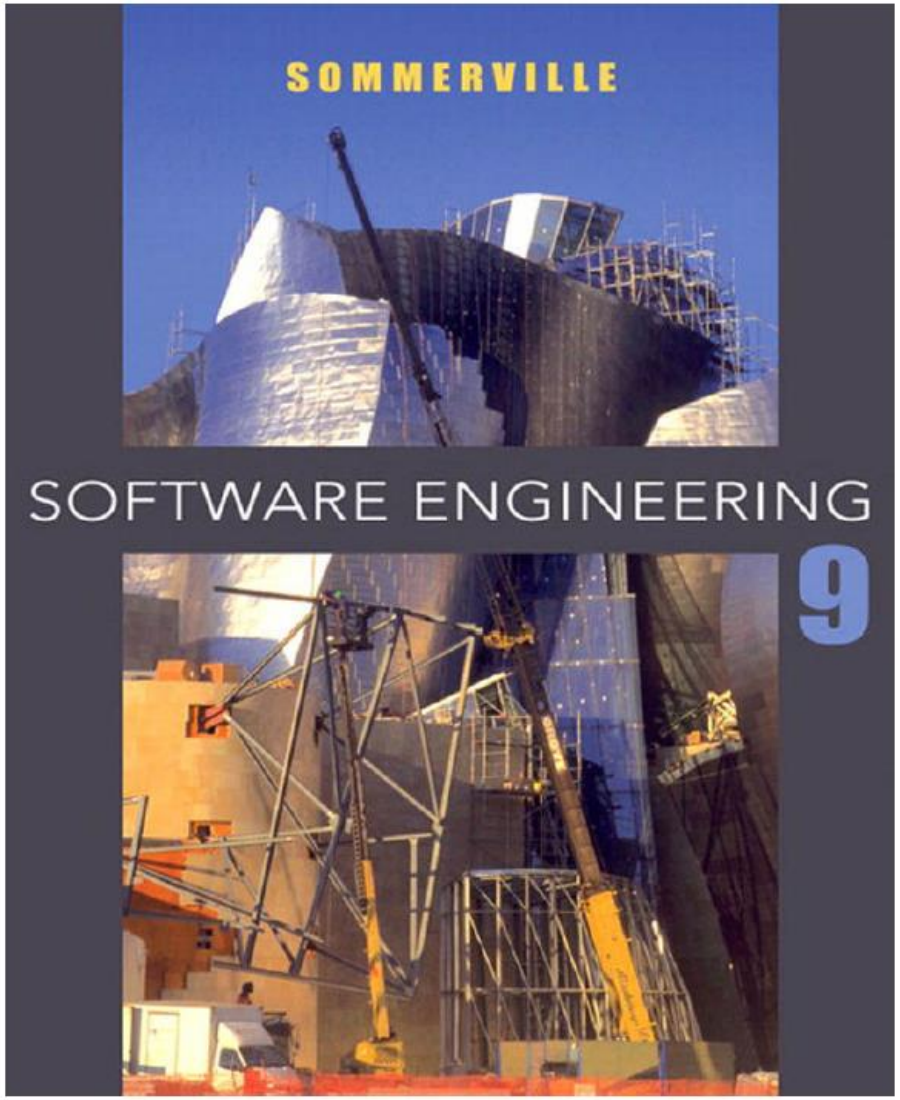
Giriş

Ders Günü ve Saati:
Salı: 09:15-13:00
Devam zorunluluğu %70



HAFTA	KONULAR
Hafta 1	Yazılım Mühendisliğine Giriş
Hafta 2	Yazılım Geliştirme Süreç Modelleri
Hafta 3	Yazılım Gereksinim Mühendisliği
Hafta 4	Yazılım Mimarisi
Hafta 5	Nesneye Yönelik Analiz ve Tasarım
Hafta 6	Laboratuar Çalışması: UML Modelleme Araçları
Hafta 7	Yazılım Test Teknikleri
Hafta 8	Ara Sınav
Hafta 9	Yazılım Kalite Yönetimi
Hafta 10	Yazılım Bakımı - Yeniden Kullanımı ve Konfigürasyon Yönetimi
Hafta 11	Yazılım Proje Yönetimi (Yazılım Ölçümü ve Yazılım Proje Maliyet Tahmin Yöntemleri)
Hafta 12	Yazılım Proje Yönetimi (Yazılım Risk Yönetimi)
Hafta 13	Çevik Yazılım Geliştirme Süreç Modelleri
Hafta 14	Yazılım Süreci İyileştirme, Yeterlilik Modeli (CMM)

Kaynaklar



6.
Yazılım İnceleme

Giriş

BÖLÜM HEDEFLERİ

- Yazılım Kalite değerlendirmede incelemenin (inspection) yerini anlama
- Test ve İncelemenin farkını anlama
- İnceleme ekibi yapısı ve rollerini öğrenme
- Teknik İnceleme Raporu

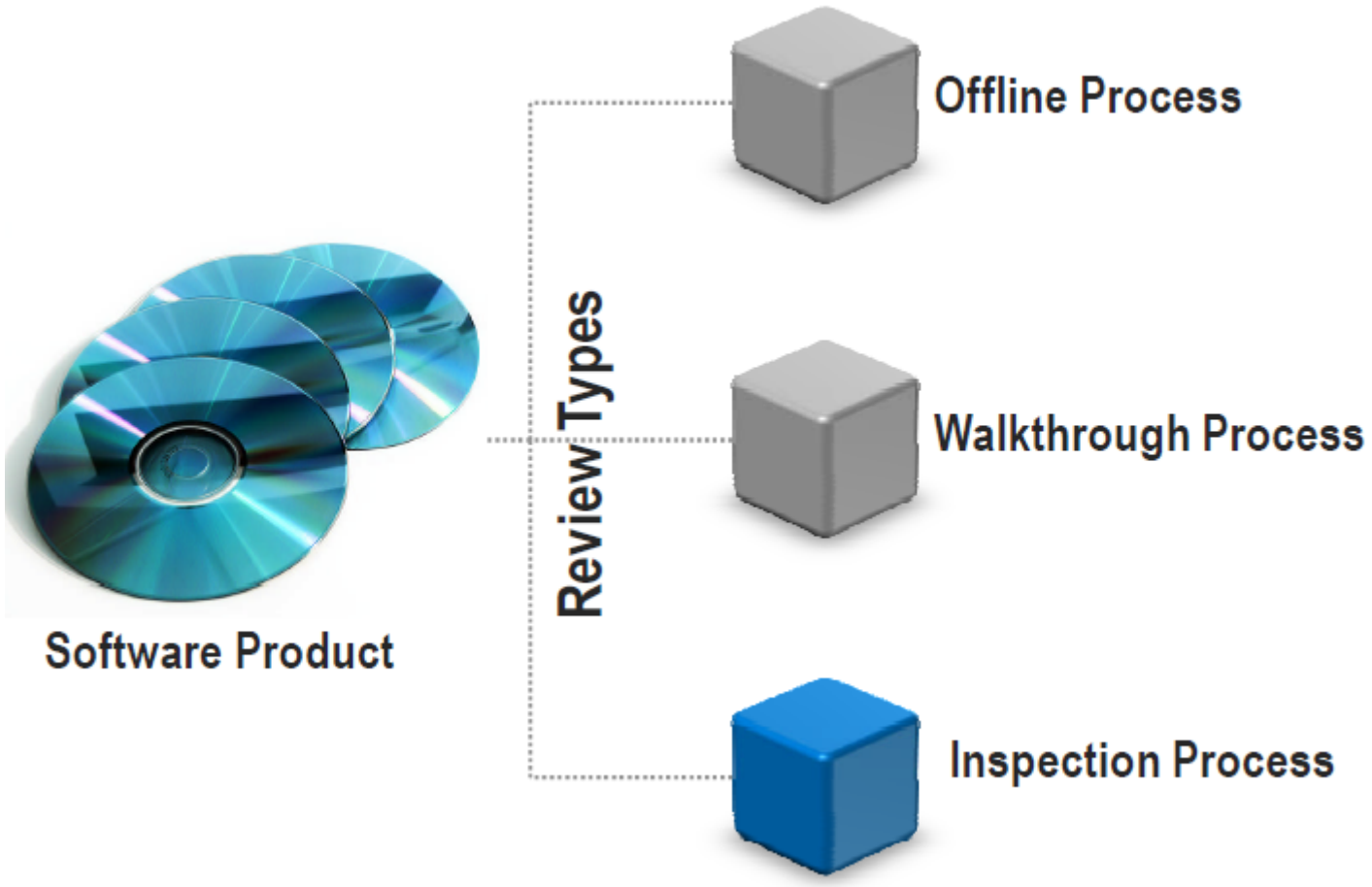
HAFTA İÇERİĞİ

- Yazılım inceleme temel kavramlar.
- Yazılım incelemenin önemi
- Genel yazılım inceleme süreci
- Fagan İncelemesi
- Diğer İncelemeler
- İnceleme ekibi
- İnceleme özet raporu

Yazılım inceleme temel kavramlar.

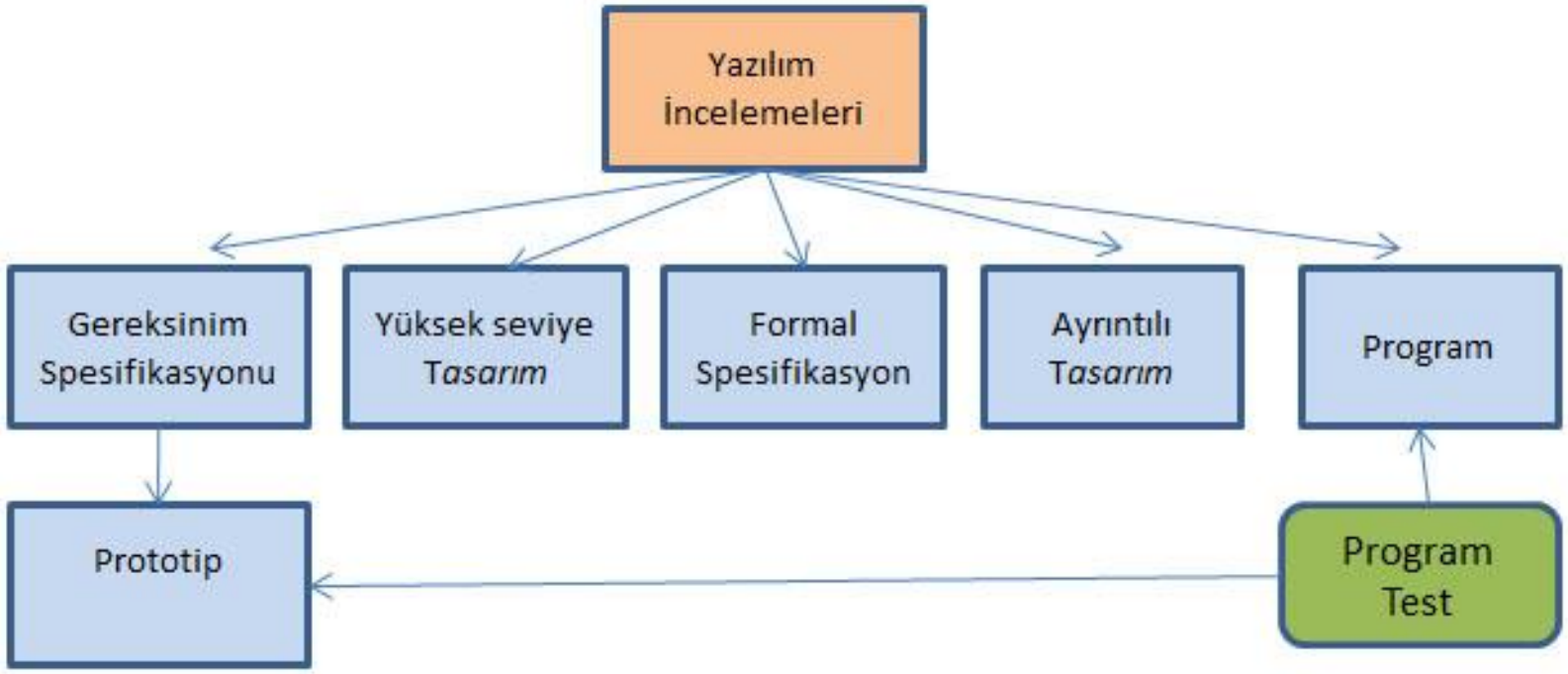
- Yazılım incelemesi, test yöntemiyle birlikte yazılım kalitesinin sağlanması için en çok kullanılan yöntemdir.
- Temel nedenler; nitelikli personel gereksinimi, biçimsel uygulama ve sürecin sonlanma aşamasının (başarısının) net olarak tanımlanamaması olarak verilebilir.
- Yazılım inceleme çalışmaları belgeleri doğrulamak için maliyetli, fakat bir o kadar da başarılı bir yöntemdir. İlk olarak Michael Fagan tarafından 1976 yılında tanımlanmış, ve genel olarak kabul görmüş bir yöntemdir.
- İnceleme, statik test aşamasında kabul edilen en resmi gözden geçirme biçimidir.

Yazılım inceleme temel kavramlar.



Yazılım inceleme temel kavramlar.

- Test yönteminin aksine hataları kodları çalıştırmadan tespit etme esasına dayanır.
- Statik geçerleme yöntemidir



Yazılım inceleme temel kavramlar.

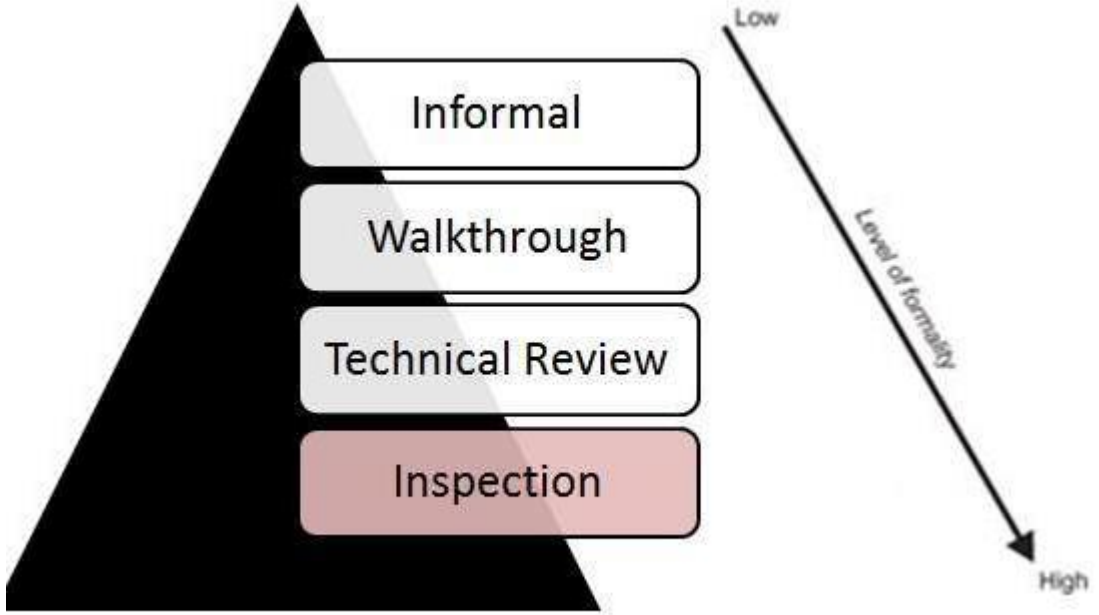
- Test, inceleme yazılım kalitesini yönetmek için en çok kullanılan yaklaşımdır
- Test ve inceleme, gerçek tasarım ve kodlamadan genellikle daha fazla kaynak kullanımı için geçerlidir.
- Test ve inceleme tüm kusurları bulamaz
- Test ve inceleme kalitesi oluşturmaz. Geliştirme uygulamaları kaliteyi yaratır
- Test ve inceleme ile ilgili seçenekler, yönetim tarafından görülebilen kalite ölçütlerinden etkilenir.

Yazılım inceleme temel kavramlar.

- Yazılım sistemlerinin içerdiği hataların insanlar tarafından incelenerek tespit edilmesi esasına dayanan bu süreçte insan faktörü çok önemli bir rol oynar. Yazılım sistemlerini inceleyen insanların bilgi ve tecrübeleri bu süreçte ana unsurdur.
- Yazılım inceleme çalışmaları literatürde genel olarak üç farklı tanımla karşımıza çıkmaktadır.
  1. İlki tasarım inceleme olarak ele alınmakta, tasarım ve ilgili belgelerin incelenmesini kapsamaktadır.
  2. İkincisi kod inceleme çalışmasıdır ve kaynak kodun incelenmesini kapsamaktadır.
  3. Üçüncüsü ise Fagan inceleme olarak bilinmektedir.

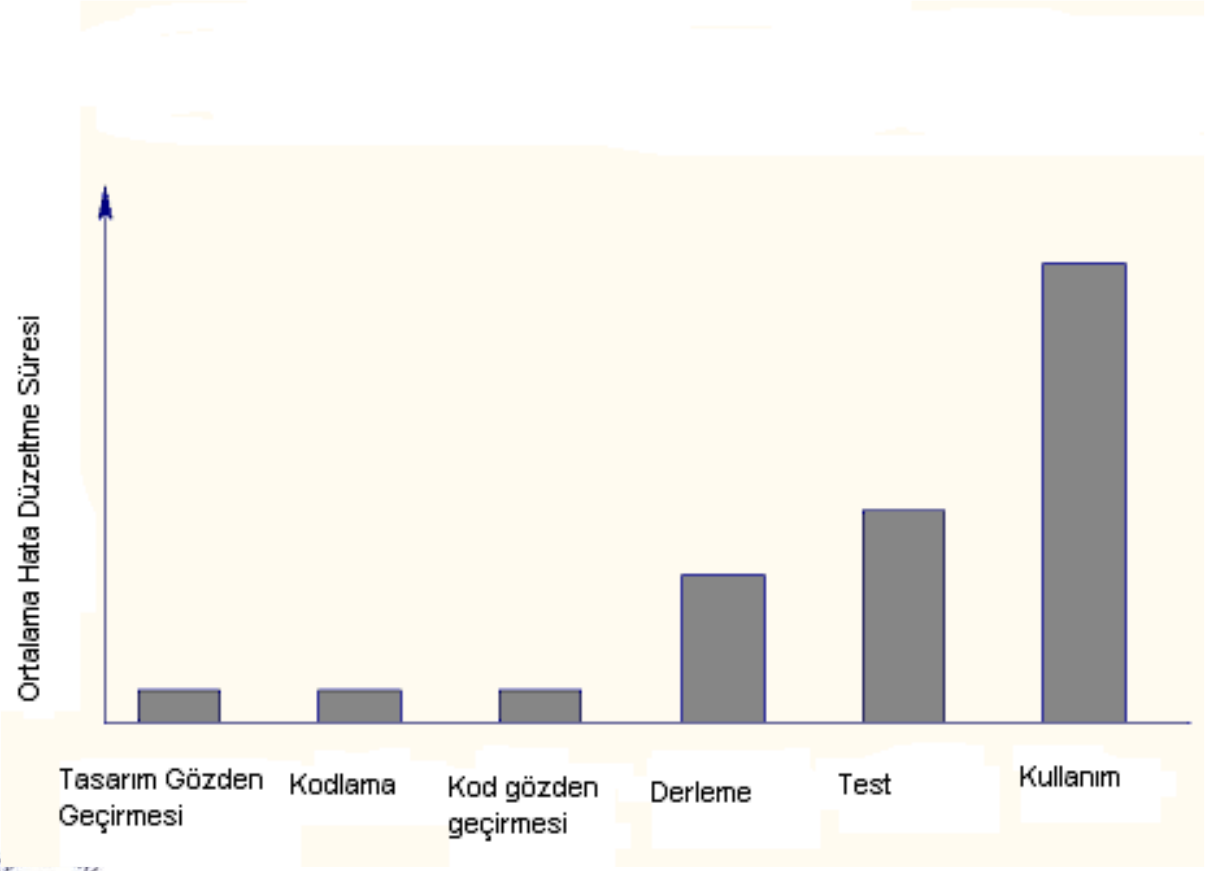
Yazılım inceleme temel kavramlar.

- IEEE(1028-1997)[11] standardına göre(Fagan Yazılım İnceleme Modeliyle aynıdır) yazılım inceleme çalışmaları proje planına uygun olarak yapılmalıdır. Ayrıca yazılım doğrulama ve geçerli kılma planında, yazılım inceleme çalışmalarının nerelere yansıtılacağı belirlenmelidir. IEEE standardına göre yazılım inceleme çalışması yazılım ürününün uygunluğunu değerlendirme çalışmasından öte, bir sınama çalışmasıdır



Yazılım incelemenin önemi

- Yazılım geliştirme sürecinin daha sonraki aşamalarında, özellikle Test ve Müşteri değerlendirmesi sırasında bulunan hataların onarılması maliyeti daha fazla etkiler.
- Yazılım incelemesi ile, yazılım geliştirme süreci süresi ve maliyeti azalır, kalite ve müşteri memnuniyeti artar.



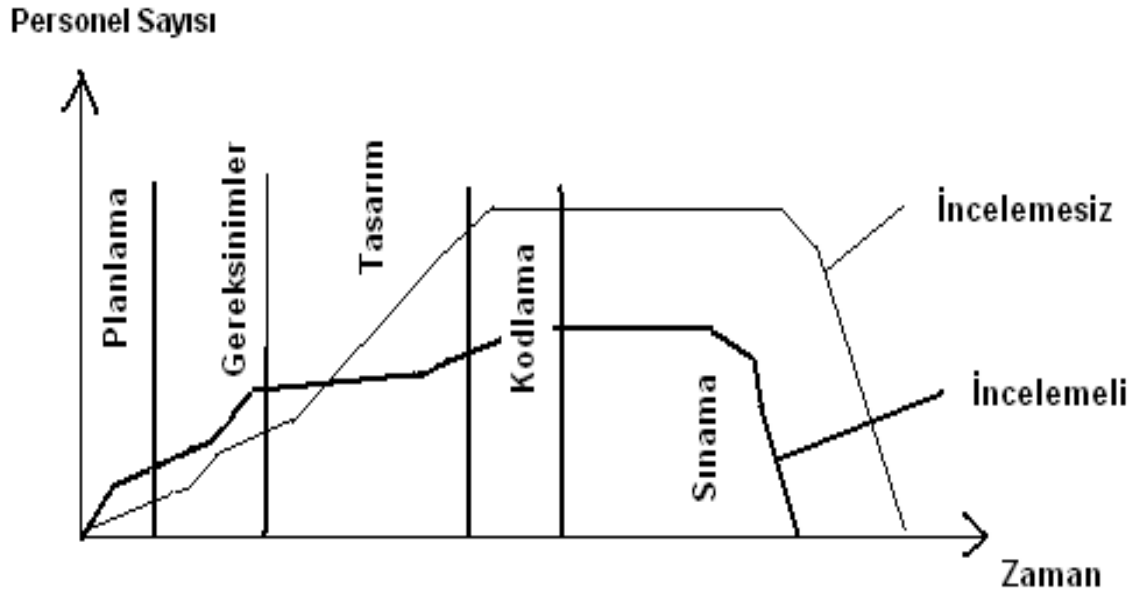
Yazılım incelemenin önemi

- Yazılım inceleme teknikleri eğer doğru uygulanabilirse, yazılım hatalarının %80'ine yakın kısmının yazılım geliştirmenin önceki safhalarında yakalanabileceğini göstermiştir.
- Boehm ve Basili kaçınılması mümkün tekrarlı işlerin %80'lik kısmına, hataların %20'sinin sebep olduğunu göstermişlerdir.. Bu amaçla hataların yazılım modelinde kodlamadan önceki süreçlerde tespit edilmesi önemlidir ve yazılım inceleme tekniği bu amaçla kullanılmalıdır.
- Hatta yazılım modelinin, yazılım inceleme çalışmalarını göz önünde bulundurarak hazırlanmasının hataların bulunması ve önlenmesinde başarılı olacağı düşünülmektedir.*

*M. Turan, F. Buzluca, YKGS2008: Yazılım Kalitesi ve Yazılım Gelistirme Araçları 2008 (9-10 ekim 2008, İstanbul)

Yazılım incelemenin önemi

- Yapılacak hazırlıklar, uygulama zamanı ve kullanılacak insan gücü göz önüne alındığında ilk başta uygulanması maliyetli gözükse de, yapılan çalışmalar kaynak tasarrufu ve ürün kalitesinin arttırılması konusunda çok ciddi kazanımlar elde edildiğini göstermiştir



Yazılım İnceleme Çalışmasının Proje Planına Etkisi(Fagan,1976)

Yazılım incelemenin önemi

- Özellikle Doolan'ın, incelemeye ayrılan her saatin, 30 mislinin yatırım olarak geri döndüğü ve Russell'in, incelemeye ayrılan her saatin, bakım aşamasında 33 saat olarak geri döndüğüne dair oldukça etkileyici çalışmaları mevcuttur.

- Kod incelemeleri, herhangi bir başka test yöntemi ile ikame edilemeyen yüksek verimli bir test yöntemidir. Zaman alıcıdır, ancak istatistiklere göre, doğru şekilde yapılırsa, içerdiği hataların% 80'ine kadarını bulacaktır. Bununla birlikte, her şey uygulanan yöntemlere ve kontrollere ve müfettişlerin titizliğine bağlıdır. Genelde birkaç saat süren tek bir toplantıda yapılan "kod gözden geçirme" ya da "geçmesi" ile karıştırılmamalıdır. Uygun bir kod incelemesi birkaç gün sürebilir ve kullandıkları yerleri bulmak için sembollere göz atan araçların yardımına ihtiyaç duyar. Yazılım yaşam döngüsünde neredeyse tüm iş ürünleri için uygun denetimler uygulanabilir. İlk bakışta çok zaman alıcı olabilirler. Ancak istatistiksel değerlendirmeler, yazılım geliştirme sürecinin tüm yaşam döngüsü boyunca, kaynakları ve dolayısıyla para tasarrufu sağladığını ve ürünün kalitesini geliştirdiğini göstermiştir.

Yazılım incelemede Karşılaşılan Güçlükler

Yazılım inceleme çalışmasının uygulanmasında karşılaşılan güçlükleri aşağıdaki gibi maddeler halinde listelemek mümkündür.

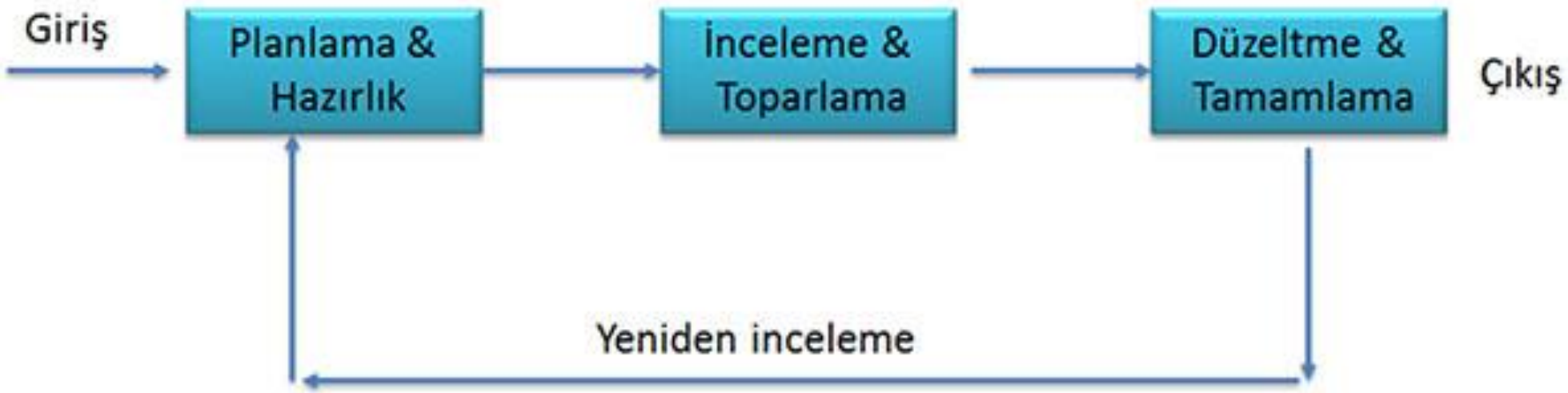
- Bir yazılım inceleme çalışmasında mevcut standartlar göz önüne alındığında an az 5 veya 6 yetkin kişinin katılması beklenmektedir.
- Yazılım inceleme çalışmasında biçimsellik çok baskındır. Biçimsellik yazılım incelemenin amaçlanan şekilde yapıldığını garanti eder ve toplantıların kurallara uygun yapılmasını sağlar. Bununla birlikte hataların yakalanacağının garantisi yoktur.

Yazılım incelemede Karşılaşılan Güçlükler

- Hazırlanma aşaması büyük önem taşımakla birlikte uygulamada belirsizlikler vardır. Yazılım incelemede dikkat edilmesi gereken noktalar hakkında ortak bir görüş veya standard bulunmamaktadır.
- Yazılım geliştirme yöntemlerinin yazılım inceleme çalışması üzerine etkilerini gösteren deneysel çalışmalar bulunmamaktadır.
- Yazılım incelemenin başarısı ve yönetim için hangi bilgisayar destekli araçların kullanılması gerektiği konusunda net bir görüş bulunmamaktadır.

Genel yazılım inceleme süreci

- Yazılım inceleme süreci genel olarak 3 aşamadan oluşur



Genel yazılım inceleme süreci

- **Planlama ve Hazırlık Aşaması:**

İnceleme ile ilgili genel soruları cevaplamayı amaçlar. Bu sorular,

- İncelemenin amaç ve hedefleri nelerdir?
- İncelenecek yazılım parçaları hangileri olacaktır?
- Bu incelemeyi kim yapacak?
- İncelemeyi yapacak kişiler hangi özellikleri taşımalılar?
- Bütün süreç boyunca hangi inceleme teknikleri ve takip eylemleri uygulanacak?

Genel yazılım inceleme süreci

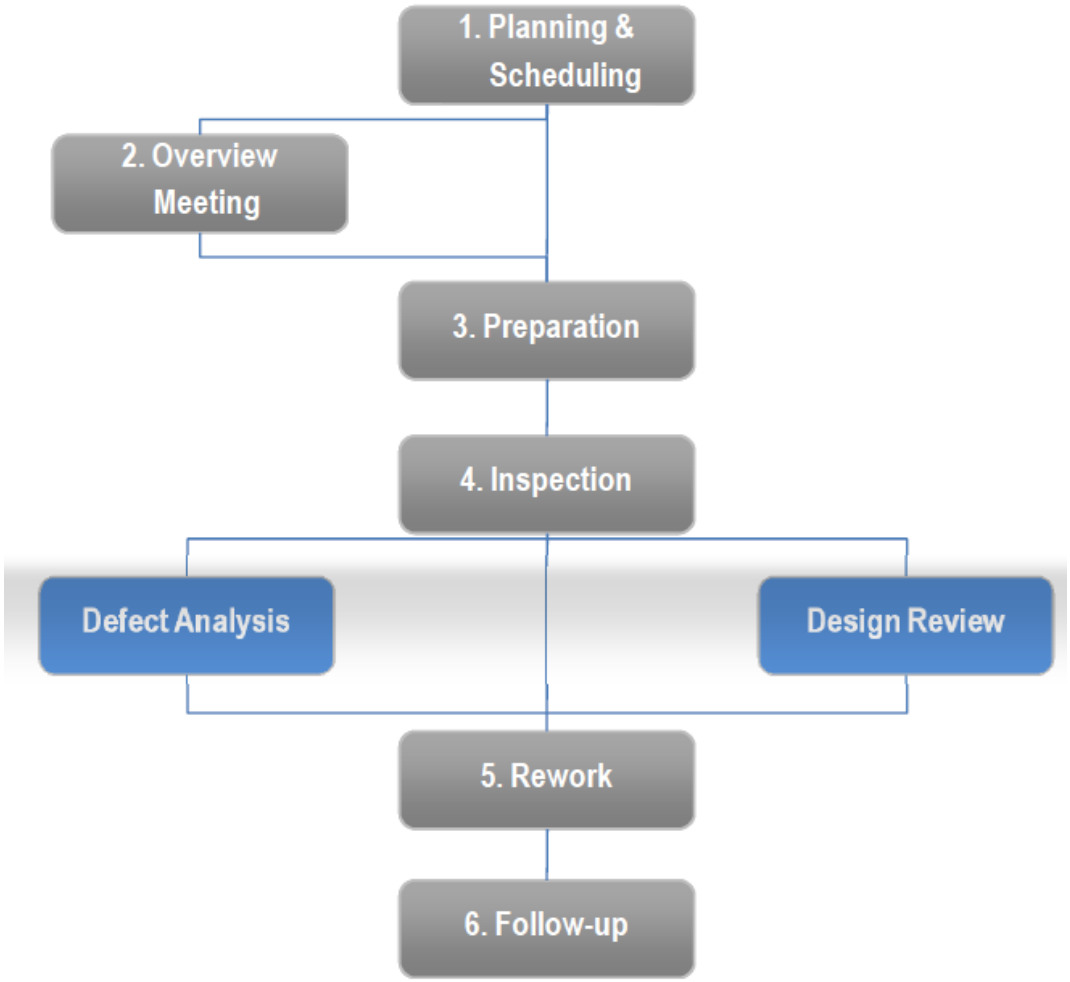
- **İnceleme ve toparlama aşaması:**

Belirlenen kriterlere uygun olarak inceleme işleminin yapılması, hataların tespit edilmesi ve bu hataların bir sonraki aşamada düzeltilmek amacıyla kayıt altına alınması.

- **Düzeltme ve tamamlama aşaması:**

Bulunan hataların, yazılımları incelemekle sorumlu kişiler tarafından düzeltilmesi ve takip edilmesidir. İstenen sonuçlar elde edilemezse yeniden inceleme için sürecin başına dönülür.

Genel yazılım inceleme süreci

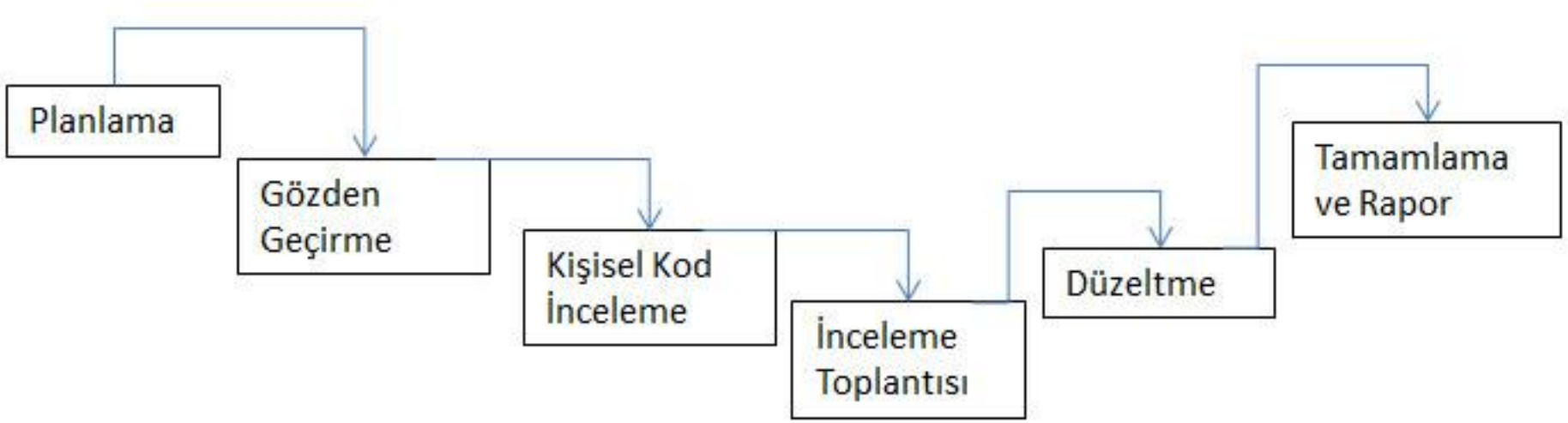


Fagan İncelemesi

- Yazılım incelemesi ile nerdeyse eş anlamlı olarak kabul edilen Fagan incelemesi bu konuda yapılmış en eski (Fagan,1976, 1986) ve en etkili çalışmadır.
- Fagan incelemesi çok geniş bir endüstriyel alanda çok çeşitli yazılımlarda program kodları üzerinde uygulanmış bir yöntemdir.
- Bu yöntemi esas alan çok değişik çalışmalar da daha sonraları yapılmıştır.
- Katalizör yöntemlerden biridir.
- Birçok yazılım kuruluşu, kendi inceleme prosedürleri için bu süreci (veya bir varyasyonu) kabul etmiştir.

Fagan İncelemesi

- Her ne kadar orjinal Fagan incelemesi 5 aşamadan oluşmuşsa da daha sonra yapılan çalışmalarda ve uygulamalarda 6 farklı aşamada değerlendirilmiş ve kabul edilmiştir



Fagan İncelemesi

- **PLANLAMA** aşamasında ne incelenecek, kim hangi rolü üstlenerek bu incelemeyi yapacak, inceleme için ortam hazır mı ve ne kadar süre gerekecek sorularının cevapları aranır.
- Denetlenecek yazılımın belirli giriş kriterlerine uygun olup olmadığını kontrol eder.
- Eğer öyleyse, genellikle en çok dört kişiden oluşan bir denetim ekibi oluşur.
- Müfettişler genellikle benzer yazılım üzerinde çalışan geliştiricilerden oluşan bir havuzdan, mevcut yapay nesnelerle arayüz yapan yazılımdan seçilir.
- Varsayım, esere aşina olan denetçilerin, olmayanlardan daha etkili olacağıdır.

Fagan İncelemesi

2- GÖZDEN GEÇİRME aşamasında kodu yazan kişi kodu inceleyecek kişilerle bir araya gelir ve onlara yazılımla ilgili bilgi verir. Kodlar inceleyecek kişiler arasında paylaştırılır.

- Yazar denetim ekibi ile toplanır. Yazılım üzerinde örneğin amacı ve diğer eserler ile ilişkisi gibi konularda arka plan sağlar,.

3.HAZIRLIK (KİŞİSEL İNCELEME) aşamasında her bir inceleyici dikkatli bir şekilde olası hatalara ve soru işareti olan kısımlara odaklanarak muhtemel hataları tespit etmeye çalışır, inceleme sorumlusuna formlar verilir.

- Denetim ekibi bağımsız olarak eseri ve destekleyici belgeleri analiz eder ve potansiyel kusurları kaydeder.

Fagan İncelemesi

4-İNCELEME toplantısında inceleyicilerin incelediği her bir kısma ait inceleme sonuçları toplanır, Kayıtçı, bu sonuçları önem derecesi ve sınıfı ile kaydeder ve inceleme sorumlusu yönetiminde değerlendirilir.

- Denetim ekibi, sadece hata bulma amacı ile yazılımı analiz etmek için toplanır. Toplantıda, birlikte çalışan bir grup insanın, tek başına çalışan üyelerin, kusurlarını buldukları varsayımı üzerine düzenlenmiştir.
- Toplantıdan önce, toplantıyı yöneten bir kişi takım lideri veya moderatör olarak belirlenir. Okur olarak belirlenmiş bir başka kişi, artefaktı işaret eder. Hatalar okuyucunun söylemi sırasında bulunur ve sorular sadece kusurların tanındığı noktaya kadar takip edilir. Bulunan konular bir denetim raporunda belirtilmiştir ve yazarın bunları çözmesi gerekmektedir. (İnceleme sırasında kapsamlı çözüm avcılığı önerilmez.) Denetim toplantısı, yorulmayı önlemek için 2 saatten fazla sürmez.

Fagan İncelemesi

5-DÜZELTME aşamasında kodu yazan kişi (yazar) inceleyicilerin yaptığı çalışmaları dikkate alarak kodlarda gerekli değişiklikleri yapar.

- İnceleme raporunda belirtilen tüm konular yazar tarafından çözülür.

6-TAMAMLAMA VE RAPOR aşamasında, inceleme süreci son bir doğrulama ile tamamlanır, yazılım kalite grubuna iletilir.

- Her bir sorunun çözümü, moderatör tarafından doğrulanır. Moderatör daha sonra, çalışmanın miktarına ve kalitesine bağlı olarak eserin yeniden değerlendirilip değerlendirilmeyeceğine karar verir.

Diğer İncelemeler

Yazılım incelemesi ile ilgili birbiriyle iç içe geçmiş çok fazla teknik bulunmaktadır. bu konuyla ilgili belli başlı ve kabul görmüş yöntemler vardır. Bu teknik ve süreçler,

- İnceleme kapsamı ve alanı
- İnceleme ekibi büyüklüğü ve
- İnceleme kurallarına göre

belirlenmiştir.

Diğer İncelemeler

Kapsamı ve Ekip Büyüklüğü Azaltılmış inceleme

Standart Fagan incelemesinde ekipler grup incelemesinin faydalarından yararlanmak ve daha kolay hata tespiti yapabilmek amacıyla 4 kişiden oluşmaktadır. Bazı durumlarda yazılım incelemesi için bir veya iki kişi yeterli olmaktadır. Bu tip projelerde inceleme grupları azaltılarak projeye uygun hale getirilir.

Bisant ve Lyle, denetim ekibini iki kişiye indirmeyi önermiştir: yazar ve bir yorumcu. Bir yorumcudan oluşan küçük bir ekip kullanır;

Bir seans vardır: hazırlık, ad hoc tekniklerini kullanır; ve tek yorumcu ile yazar arasında bir toplantı yapılır.

Diğer İncelemeler

Kapsamı ve Ekip Büyüklüğü Arttırılmış İnceleme

Standart Fagan incelemesinde ekipler grup incelemesinin faydalarından yararlanmak ve daha kolay hata tespiti yapabilmek amacıyla 4 kişiden oluşmaktadır. Bazı durumlarda yazılım incelemesi için 4 kişi yeterli olmamaktadır. Projenin büyüklüğüne göre bu sayı 6 ve üzeri rakamlara çıkabilmektedir. Sayı artırılarak projeye uygun şekilde görev paylaşımı yapılır

Diğer İncelemeler

Gilb İncelemesi

Gilb incelemeleri, Fagan incelemelerine benzer, ancak Gilb'in süreci, denetim toplantısından hemen sonra bir süreç beyin fırtınası toplantısı önerir. Bu adım, denetimde bulunan kusurların nedenlerini araştırmak ve tartışmak suretiyle gelecekteki ortadan kaldırılması için olumlu tavsiyelerde bulunarak süreç iyileştirmesini mümkün kılar. Bu öneriler, geliştiricilerin çalıştığı teknik, organizasyonel ve politik ortamı etkileyebilir.

Diğer İncelemeler

Toplantısız İnceleme

Birçok kişi denetim toplantısında çoğu kusurun tespit edildiğine inanmaktadır. Bununla birlikte, bazı yeni çalışmalar, çoğu kusurun aslında hazırlık aşamasında bulunduğunu göstermiştir. Humphrey "iyi yönetilen denetimlerde bulunan hataların dörtte üçünün hazırlanma sırasında bulunduğunu" belirtmektedir. Votta, denetim toplantılarının depozitlerle değiştirilmesini önermektedir; yazar ve isteğe bağlı olarak, moderatör her biri ile ayrı ayrı toplanmaktadır. İnceleme sonuçlarını almak için gözden geçirenler.

Diğer İncelemeler

Kod Okuma

Etkili bir kod okuma tekniği hem yukarıdan-aşağıya hem de aşağıdan –yukarıya incelemenin bir birleşimi olmalıdır. Program mümkün olan en küçük parçalara (örneğin fonksiyon, prosedür, altprogram) ayrılarak öncelikle bu parçaların kendi içindeki bütünlüğü ve doğruluğu incelenmeli sonra da programın bütünü ile olan uyumları incelenmelidir.

Kod okuma, resmi kod denetimlerine alternatif olarak önerilmiştir [33]. Kod okumada, müfettiş sadece kaynak kodunu okumaya ve kusurları aramaya odaklanır. Yazar, kaynak listelerini (1K - 10K satırları), kodu günde 1 K satırlık tipik bir hızda okuyan iki veya daha fazla denetçiye verir. Bu ana adımdır. Müfettişler daha sonra kusurları tartışmak için yazarla buluşabilir, ancak bu isteğe bağlıdır. Toplantılardaki vurgunun kaldırılması, bireysel kusur keşfine daha fazla önem verilmesini sağlar. Ayrıca, toplantılarla ilgili sorunlar otomatik olarak ortadan kalkar (planlama zorlukları ve yetersiz hava süresi dahil).

Diğer İncelemeler

EXAMPLE INSPECTION METHODS (This table compares the example inspection methods based on the inspection taxonomy.)

Method	Team Size	No. of Sessions	Detection Method	Meet	Post
Fagan [15]	Large	1	Ad hoc	Yes	—
Bisant	Small	1	Ad hoc	Yes	—
Gilb [20]	Large	1	Checklist	Yes	Root cause analysis
Meetingless inspection [47]	Large	1	Unspecified	No	—
ADR [35]	Small	>1 Parallel	Scenario	Yes	—
Britcher [6]	Unspecified	4 Parallel	Scenario	Yes	—
Phased inspection [27]	Small	>1 Sequential	Checklist (comp)	Yes (reconcile)	—
N-fold [44]	Small	>1 Parallel	Ad hoc	Yes	—
Code reading [33]	Small	1	Ad hoc	Optional	—

İnceleme ekibi

- IEEE standardı, denetim için aşağıdaki rollerin belirleneceğini belirtmektedir:
  - Denetim lideri
  - Ses kayıt cihazı
  - Okuyucu
  - Yazar
  - Denetimciler

İnceleme ekibi

- İnceleme Ekibi, İnceleme Sorumlusu - İnceleyiciler - Yazar – Kod Okuyucu ve Kayıtçı dan oluşmaktadır.
- İnceleme Sorumlusu, katılımcıları belirler ve toplantıyı planlar ve yönetir, Yazar tarafından toplantı sonrasında yapılan düzeltmeleri kontrol eder, inceleme sonuçlarını yazılım kalite grubuna gönderir.
- İnceleyiciler, Hata aramak üzere yazılım ürününü inceler ve raporlar.
- Yazar, Yazılım ürününü hazılayandır, toplantıda soruları yanıtlar, toplantı sonrasında ise hataları düzeltir.
- Okuyucu, belgeyi satır satır okur yazılım ürünü incelemesinde hangi sayfa ve satırda kalındığını izler. Gilb ve Graham(1993) tarafından Yenilenmiş değerlendirmede inceleme ekibinde bulunmaması önerilmektedir.
- Kayıtçı –Bulunan hataları önem seviyesini belirleyerek kaydeder .

İnceleme ekibi

Kalite Yöneticisi	Çalışmaların kalite planına uygun olup olmadığını denetler. Ufak ölçekli projelerde kalite yönetimini proje yöneticisi yerine getirebilir.
Proje Yöneticisi	Projenin durumunu takip etmek,gerektiğinde öneri ve görüşlerini belirtmek üzere inceleme çalışmalarında yer alır.
Sınama Ekip Temsilcisi	Sınama müdürü , sınama ekip yöneticisi veya sınama ekibinden görevli bir personel sınama çalışmalarını değerlendirmek üzere inceleme çalışmalarında bulunur. Ufak ekiplerde bu görevi bir yazılımcı üstlenebilir.
İnceleyici (ler)	İnceleme yapılan konuda(analiz,tasarım veya kodlama) yetkin ve tecrübeli personel/personellerden oluşur.
Diğer Ekip Elemanları	İhtiyaç duyulan(net olmayan hususları açıklamak üzere) toplantılara yazılım ürününü üreten ekipten temsilciler katılabilir.

İnceleme ekibi

- Yukarıda yer alan görevler göz önüne alınırsa ufak ölçekli projelerde proje yöneticisi, sınamadan görevli bir personel ve bir inceleyici olmak üzere üç kişi yeterli gözükmektedir. Bu modelin, IEEE standartlarında tanımlanan biçimsel ve zorlayıcı ekip yapısına göre uygulanabilirliği oldukça fazladır.

Teknik İnceleme Özet Raporu

- Gözden geçirme; inceleme grubu üyeleri tarafından tek tek yazılım öğeleri (örneğin: gereksinimlerin spesifikasyonları, modül veya küçük modül gruplarının tasarımları, bir modülün kaynak program listesi) üzerinde ve bağımsız olarak gerçekleştirilmektedir.
- Belirli bir durak noktasındaki incelemeler tamamlanınca, geliştirme ve inceleme grupları bir arada toplanarak, gözden geçirme işlemini sonuçlandırmaktadırlar.
- Bu toplantıda bir üye raportör olarak görevlendirilmekte ve bir "teknik inceleme özet raporu" düzenlenmektedir.

Teknik İnceleme Özet Raporu

Bu raporda;

- Neyin gözden geçirildiği
- Kimin gözden geçirdiği
- Hangi hata ve eksiklerin bulunduğu ve ne yapılması gerektiği

Sonuç olarak da:

A. Değişiklik yapılmaksızın kabul edilmesi,

B. Belirtilen önemsiz hataların düzeltilmesi koşulu ile kabul edilmesi,

C. Reddedilerek düzeltme sonunda yeniden incelenmesi

kararlarından biri verilmektedir.

GÖZDEN GEÇİRME – İNCELEME KARŞILAŞTIRMASI

Konu	Teknik Gözden Geçirme	İnceleme
Amaç	Bir iş ürününün kullanım amacı için uygunluğunu belirleme.	Bir iş ürününün kullanım amacı için uygunluğunu belirlemek, ancak bu incelemenin ötesinde, eğitimli müfettişler aracılığıyla inceleme yoluyla anormallikleri araştırmak.
Roller	En az iki kişi gereklidir. Her biri birden fazla rol üstlenebilir. Kapsam farklı kişiler olduğu için örn. yönetim veya müşteri temsilcisi katılabilir, ancak bu süreçte bir rol olarak görülmemektedir.	Ek olarak gerekli roller Yazar ve Okuyucu. Roller açıkça ayrılır ve bir kişi tarafından kabul edilemez.

GÖZDEN GEÇİRME – İNCELEME KARŞILAŞTIRMASI

Konu	Teknik Gözden Geçirme	İnceleme
Girdi	Girdiler, inclemelere göre bir birine daha çok benzer. Kontrol listelerinin belirtilmediği gözlemlenmelidir.	Ek girişler şunlardır: Denetim raporlama formları, Denetim kontrol listeleri, Donanım ürün özellikleri, Donanım performans verileri. Bu girişlerin bazıları isteğe bağlıdır. Raporlama formunun yanı sıra denetim kontrol listesi de zorunludur. Standardın ifadesi, kontrol listelerinin zorunlu olmadığını göstermektedir, ancak standardın ekinde yer alan bir tablo onları zorunlu kılmaktadır..
Çıktı	Tek çıktı, bir Eylem Öğesi Listesi ve teknik incelemenin kaydıdır (toplantı tutanağı).	Çıktılar resmi bir denetim raporu, resmi bir kusur özeti ve sınıflandırılmış kusurları olan bir kusur listesidir. Vurgu, istatistiksel değerlendirmelere de izin verecek standart hata bulma çıktıları sunmaktır.

GÖZDEN GEÇİRME – İNCELEME KARŞILAŞTIRMASI

Konu	Teknik Gözden Geçirme	İnceleme
Toplantı	Gözden geçirme toplantısının açıklanan kurallarına uyulmalıdır	Tanımlanan roller açıkça tutulmalıdır.
Toplantının Sonucu	Tanımlanmış gözden geçirme raporunun oluşturulması.	İncelemenin sonunda, iş ürününü kabul etmek, denetimi kapatmak, ancak bir yeniden çalışma gerektirmek, iş ürününü reddetmek ve yeniden yapılanma sonrasında yeniden inceleme yapılmasını istemek için karar verilmelidir.

Kaynaklar

- M. Turan, F. Buzluca, YKGS2008: Yazılım Kalitesi ve Yazılım Gelistirme Araçları 2008 (9-10 ekim 2008, İstanbul)
- AdamPorter, Harvey Siy, Lawrence Votta, Advances in Computers, Volume 42, 1996, Pages 39-76

Giriş

BÖLÜM HEDEFLERİ

- Dinamik Geçerleme (verification), yazılım test sürecini tanımlama
- Birim test ve Bütünlük test işlemlerini özetlenmesi
- Regresyon testini tanıma
- Saydam kutu Kara kutu Test tiplerini inceleme
- Performans, Dayanıklılık ve Güvenlik Testi olarak Sistem Testini tanımlama

HAFTA İÇERİĞİ

- Yazılım Sınama
- Yazılım Test Süreci
- Yazılım Test teknikleri
- Hata giderme (debuging)

Yazılım Sınama

- Yazılımın, sınırlı sayıda ve seçilmiş sınama senaryoları kullanılarak beklenilen davranışı gösterdiğinin yazılımın çalıştırılması ile dinamik olarak doğrulanmasıdır.
- Yazılım sınama süreçleri, yazılımın ilk aşamalarından başlayarak gelişim süresince yazılımın var olan hatalarını ortaya çıkarmak, bunların düzeltilmesini sağlayarak ortaya konulacak olan ürünün hatasız ve isterlere uygun şekilde çalıştığını gösterme amacını içeren faaliyetlerdir.
- Ürün olarak teslim edilen sürümde sıfır hata olması pek mümkün değildir; ama önemli olan kritik hataların teslim edilen üründen arındırılmış olmasıdır.

Yazılım Sınama

Sınama (testing); bir programdaki hataları bulmak amacı ile yapılan işlemlerdir. Sınama, yazılımın;

A. fonksiyonel,
B. performans,
C. dayanıklılık,
D. yapısal

bakımlardan yeterliği denetlemek ile yürütülmektedir.

Yazılım Sınama

A. FONKSİYONEL SINAMA

Tipik bir işletme ortamında örnek girdi değerleri ile çıktının alınması ve beklenen sonuç ile karşılaştırılması biçiminde yürütülmektedir. Burada girdi olarak aşırı değerler de (en büyük ve en küçük veya negatif) verilerek, fonksiyonel sınırlar da denetlenmelidir..

B. PERFORMANSIN SINANMASI

Değişik durumlardaki yanıt süresi, altprogramların işlem süresindeki payları, iç ve dış belleklerin kullanımı, veri kanalları ve bilişim hatlarındaki iletişim hızı incelenmektedir. Böylece sistemin darboğazları belirlenmektedir.

Yazılım Sınama

C. DAYANIKLIK TESTİ

Sisteme gereğinden çok terminal bağlamak, girdi ve tanıyıcı yüklemek, bir iletişim hattını devreden çıkarmak vb. yollarla gerçekleştirilmektedir. Böylece sistemin zayıf ve kuvvetli tarafları belirlenmekte, aksama hallerinin düzeltilme olanakları saptanmaktadır.

D. YAPSIAL DENETİM

Yazılım sisteminin iç işletim mantığı sınanmaktadır. Belirli altprogramlar çağırılmakta ve bunların mantıksal yolları izlenerek, incelenmektedir

Yazılım Sınama

- Sistem üzerinde sırasıyla fonksiyonel performans ve dayanıklılık testleri uygulanmakta ve bunları tamamlamak üzere de yapısal test gerçekleştirilmektedir. Yapısal testin tasarımı, sistem geliştirildikten sonra ve önceden hazırlanan test plânına göre düzenlenmektedir.
- Fonksiyonel-performans ve dayanıklılık denetimlerine, sistemin dış spesifikasyonlarına ve gereksinimlerine dayandırıldığı için, **kara kutu testi (black box testing)** adı verilmektedir. Buna karşılık, yapısal denetimde modül düzeyinde programın deyimleri ya da dalları sınanarak iç yapısı incelenmektedir. Bu şekilde uygulanan sınama yöntemine de **saydam kutu testi (white box, glass box testing)** denilmektedir.

Yazılım Sınama

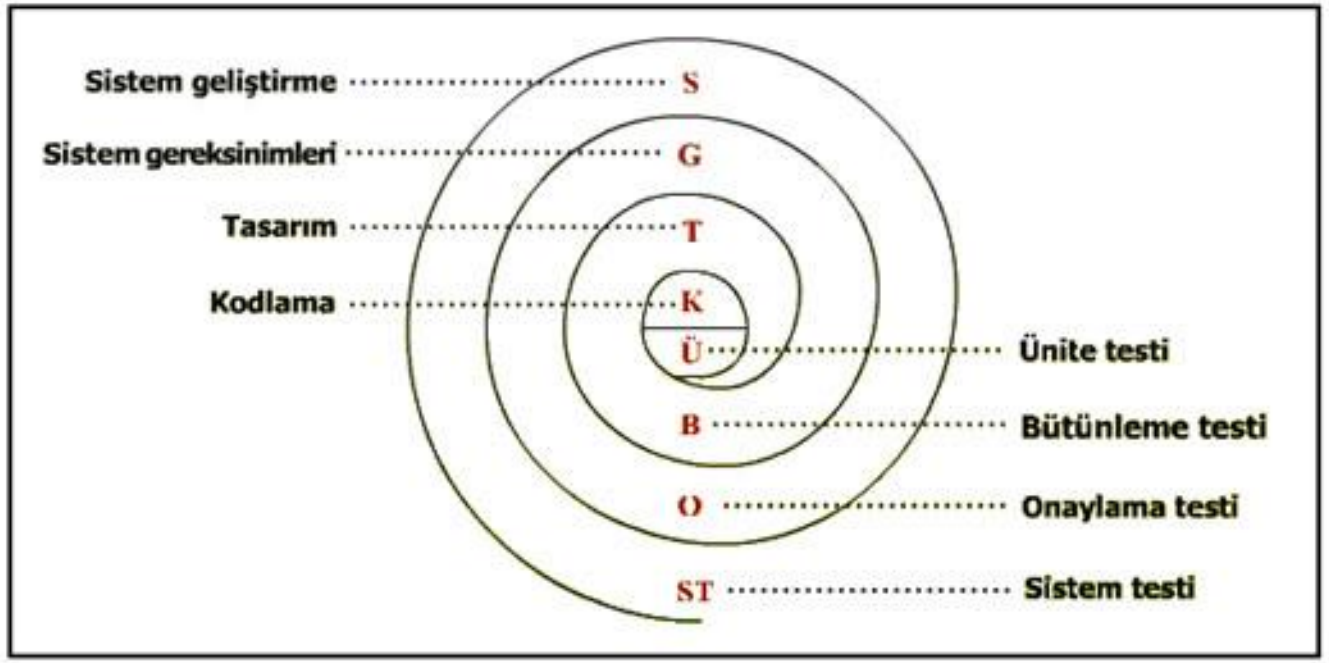
- Yazılım sınamalarının yapılması yazılım kalitesini artırmak, bakım ve gözden geçirme maliyetlerini azaltmak için gereklidir. Ayrıca, hataların bulunma zamanları ile düzeltilme maliyetleri arasında ilişki vardır;. sınamalar, yazılımdaki hataların erken fazlarda bulunmasını, bu sayede düzeltilme maliyetlerinin azalmasını sağlar.

Yazılım Sınama

- Yazılım sınama süreci genel olarak sınama hazırlık süreci, dinamik sınama süreci ve sınamaların sonlandırılması kısımlarından oluşur.
- Sınama hazırlık sürecinde; gereksinim analizi, yazılıma ait tasarım aşamalarında hangi sınamaların yapılacağı, sınamaların ne şekilde sonlandırılacağı ve dinamik sınamaların nasıl uygulanacağı ile ilgili planlar hazırlanır.
- Dinamik sınama süreci, kodlamanın bitmesine yakın bir dönemde başlar, yazılımın türüne ve özelliklerine göre uygulanacak yöntemler farklılık gösterebilir.

Yazılım Sınama

Dinamik sınama süreci; ünite testi, bütünleme testi, onaylama testi ve sistem testi basamakları halinde gerçekleştirilmektedir. Böylece sınama işlemi, yazılım geliştirmenin son basamağı olan kodlamadan geriye doğru yürütülmektedir.



BİRİM TESTİ

- Birim sınamaları, hataların erken bulunup düzeltilebilmesi için oldukça büyük öneme sahiptir ve dinamik sınama sürecinin ilk aşamasıdır.
- Ünite (birim) testi, yazılım tasarımının en küçük birimi olan modül üzerinde uygulanmaktadır. Ayrıntılı tasarım tanımlarına dayanılarak, modül içerisindeki hataları bulmak üzere, önemli kontrol yolları sınanmaktadır. Saydam kutu testi olarak uygulanan bu işlem, çok sayıdaki modül üzerinde, paralel olarak yürütülmektedir.

BİRİM TESTİ

- Kaynak kodunun iç yapısı ve tasarım biçiminin bilinmesi gerektiğinden geliştiriciler tarafından yapılır. Sınama gereksinimleri çıkarıldıktan sonra sınama planlaması yapılır.
- Ardından, yazılımın alt birimleri ve bunların bileşenlerinden oluşan parçaların hangi sırayla ve nasıl sınanacağı belirlenir.
- Yazılmış olan kaynak kodun içerdiği alt bileşenler olan sınıfların, rutinlerin, kullanılan veri yapılarının doğruluğu sınanmış olur. Bu sayede ileriki sınama süreçleri öncesi, birimlerin düzgün çalışması da sınanmış olur.

BİRİM TESTİ

- Birim testinde; modülün arabirim, lokal veri yapısı, kontrol yapıları arasındaki ana yollar, hata arama yolları ve modül sınırları sınanmaktadır.
- Modül veri yapısının sınanmasında, algoritmik işlemlerin her basamağında geçici olarak saklanan verinin bütünlüğünü koruduğu denetlenmektedir. Bu denetim için; kayıt, başlama, değer, değişken adı, veri tipi, kuyruk listedeki boşluk ve taşmalar (underflow overflow) ve adresleme hatalarını bulacak bir test programı (test case) tasarlanmaktadır. Bu arada, modüldeki genel veri de denetlenmektedir.

BİRİM TESTİ

- Kontrol yapıları arasındaki anayolların denetiminde; hesaplama, karşılaştırma ve kontrol akışı hataları aranmaktadır. Bu amaçla da test senaryoları kullanılmaktadır.
- Hesaplama hatalarına: aritmetik işlemlerin öncelik sırasında tam sayılı bayağı kesir işlemlerinde, doğruluk derecesinde, tanımların simgesel gösterimlerinde rastlanmaktadır. Karşılaştırma ve kontrol akışı, birbirine bağlı bulunmaktadır. Genellikle kontrol akışı değişikliği, karşılaştırmadan sonra yapılmaktadır.

BİRİM TESTİ

- Test programı ile; farklı veri tiplerinin karşılaştırılması, mantıksal işlemler ve sıralamalar, eşitlikler, değişkenlerin karşılaştırılması, döngünün tamamlanması, yinelemelerde çıkış yeri, döngü değişkenleri konularındaki hatalar saptanabilmektedir.
- Modül sınırlarında genellikle: n-elemanlı ve n-boyutlu dizilerin işlenmesinde bir döngünün i'inci yinelenmesinde ya da en büyük ve en küçük değerlerde hatalar oluşmaktadır. Test senaryoları ile veri yapısının, kontrol akışının ve veri değerinin işletilmesi sırasında sınır değerler verilerek, bu hatalar bulunabilmektedir.

BİRİM TESTİ

- Ünite testi, kaynak program düzenlenerek gözden geçirilip sözdizimi hataları düzeltildikten sonra uygulanmaktadır. Önce, yazılım tasarımına dayanarak, yukarıda açıklanan hata çeşitlerini bulabilecek bir test programları hazırlanmaktadır. Bu programlardan beklenen sonuçlar da belirlenmektedir.
- Birim sınamaları sırasında, birimlerin performansı ölçülebilir ve bellek analizi de yapılabilir. Geliştirilmiş olan yazılımın mimarisi bu sınamaların yapılma zorluğunu belirleyen unsurların başında gelmektedir.

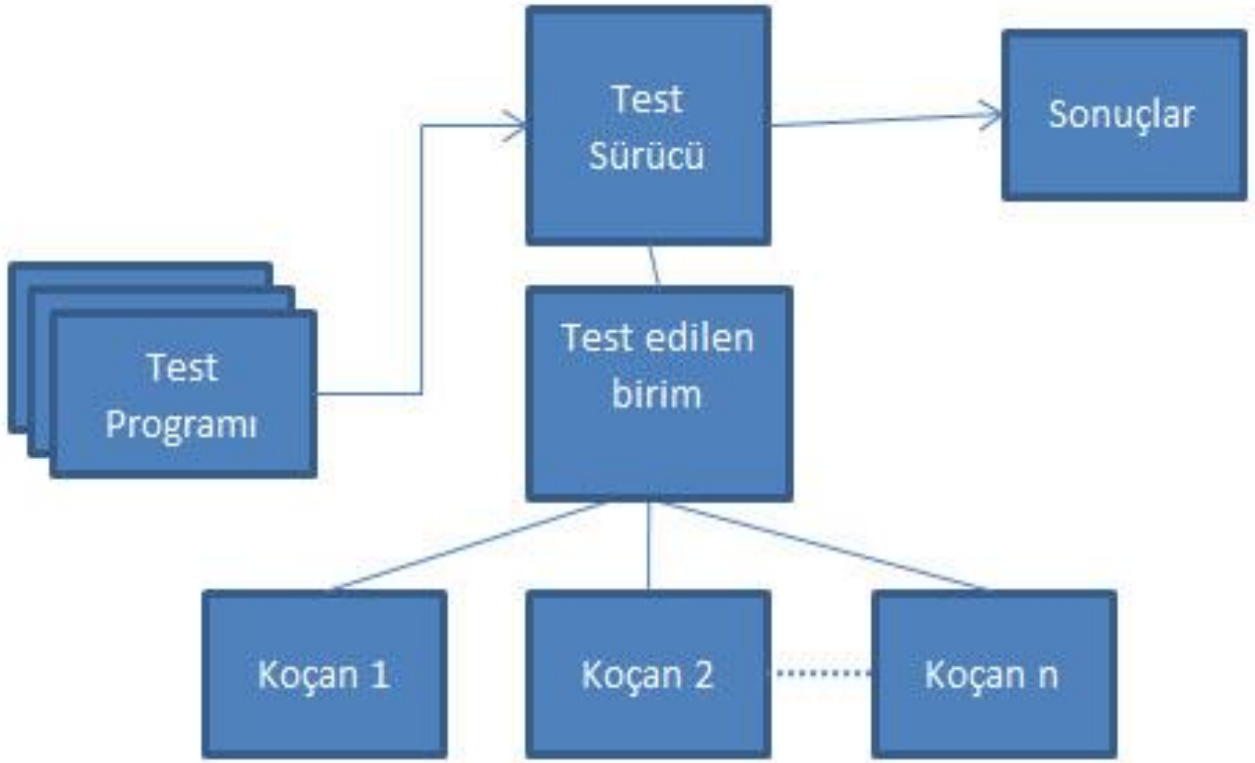
BİRİM TESTİ: TEST SENARYOSU

- Test senaryosu (test case); belirli bir program yolunu işlemek ya da özel bir gereksinime uygunluğu onaylamak amacı ile düzenlenen bir dizi sınama verisinden ve buna ilişkin işlemlerden oluşturulmaktadır. Test programlarının geliştirilmesi, diğer yazılımlar gibidir. Geliştirmeye de, test plânı uyarınca ve yazılım tasarımı ile birlikte başlanmalıdır.
- Modülün bağımsız olmaması halinde, sınamada diğer modüller de dikkate alınmalıdır. Bu amaçla her ünite testi için bir “test sürücü”(driver) ve/veya “koçan”(stub) yazılımı geliştirilmektedir.

Koçan: Bir alt bileşenin, üst bileşen ile arayüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

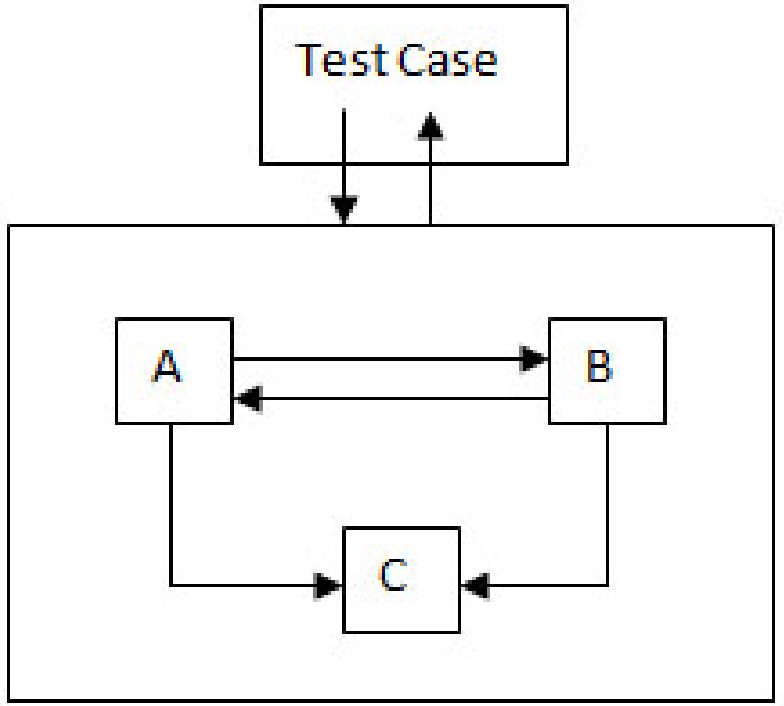
BİRİM TESTİ : TEST SENARYOSU

Test sürücü (test driver); test programı verisini alarak test edilecek modüle ileten ve test sonucunu yazan bir ara programdır. Koçan (stub); bir kukla (dummy) alt proğram olup, test edilen modülün altprogramını temsil etmektedir. Test sürücülere ve koçanlara, modül programın başlığında yer verilmelidir. Bu basamağa yetiştirilememesi halinde, tekrar kullanılacağı bütünleme testinde derlenmelidir.



BİRİM TESTİ:ARABİRİM TEST

- Daha büyük sistemler yaratmak üzere modül veya alt sistemler birleştirildiğinde arabirim testi yapılması gerekir. Her modül veya alt sistem diğer program parçaları ile arabirime sahiptir.
- Arabirim testinin amacı, arabirim hataları veya arabirimler hakkında yanlış varsayımlar nedeniyle sistemde oluşacak hataları belirlemektir.



BİRİM TESTİ:ARABİRİM TEST

Arabirim Tipleri

- **Parametre Arabirimleri:** veri veya bazen fonksiyon referanslarının bir bileşenden diğerine geçirildiği parametrelerdir.
- **Paylaşımlı Bellek Arabirimleri:** Altsistemler arasındaki bellek bloğunun paylaşıldığı arabirimlerdir.
- **İşlemsel Arabirimler:** Bir alt sistemin, diğer alt sistemler tarafından çağrılan prosedürleri sarmalayan (encapsulate) arabirimlerdir.
- **Mesaj Geçirme Arabirimler:** Bir alt sistemin bir başka alt sistemden mesaj göndererek hizmet istediği arabirimlerdir.

BİRİM TESTİ:ARABİRİM TEST

Arabirim Hataları

- **Arabirimin Yanlış Kullanımı :** Çağıran bileşen, başka bileşeni çağırır ve onun arabirimini kullanmada hata yapabilir. Parametrelerin yanlış tipde girilmesi – yanlış sırada geçirilmesi, yanlış sayıda geçirilmesi.
- **Arabirimin Anlaşılmaması:** Çağıran bileşen, çağrılan bileşenin arabiriminin özelliğini yanlış anlar. Çağırılan bileşen beklendiği gibi davranmaz.

BİRİM TESTİ:ARABİRİM TEST

Zamanlama Hataları

- Paylaşımlı bellek kullanan veya mesaj gönderen gerçek zamanlı sistemlerde olur. Veri üreticisi ve kullanıcısı farklı hızlarda işleyebilir. Arabirim tasarımında önlem alınmazsa, Bilgi üreticisi paylaşılan arabirim bilgisini güncelleyemediğinde, kullanıcı da güncel olmayan bilgiye erişecektir.

Arabirim Testi Uyarıları:

- Mesaj geçirme sistemlerinde dayanıklılık testi kullanılmalı (pratikte olabileceğinden daha fazla mesaj gönderilmeli)
- Dış birimin her bir çağırılmasında parametreleri açıkça (explicit) tanımlanmalı
- Arabirim null gösterge (pointer) parametreleri ile test edilmeli.

BÜTÜNLEME (Integration Test) Testi

- Bir uygulamanın altındaki bileşenler modüller, bağımsız uygulamalar, istemci/sunucu uygulamaları şeklinde olabilir. Bütünleme sınaması ile bu bileşenlerinin beraberce uyum içinde çalışıp çalışmadığını doğrulama için yapılır. Özellikle istemci/sunucu uygulamaları ve dağıtık sistemlerde çok kullanılmaktadır.
- Modüller bağımsız olmayıp, birbirilerine ya da bir donanım öğesine veya işlemciye bağlı olmalıdır.
- Bu bağlantı, “yazılım arabirimi” (software interface) ile sağlanmaktadır.

BÜTÜNLEME (Integration Test) Testi

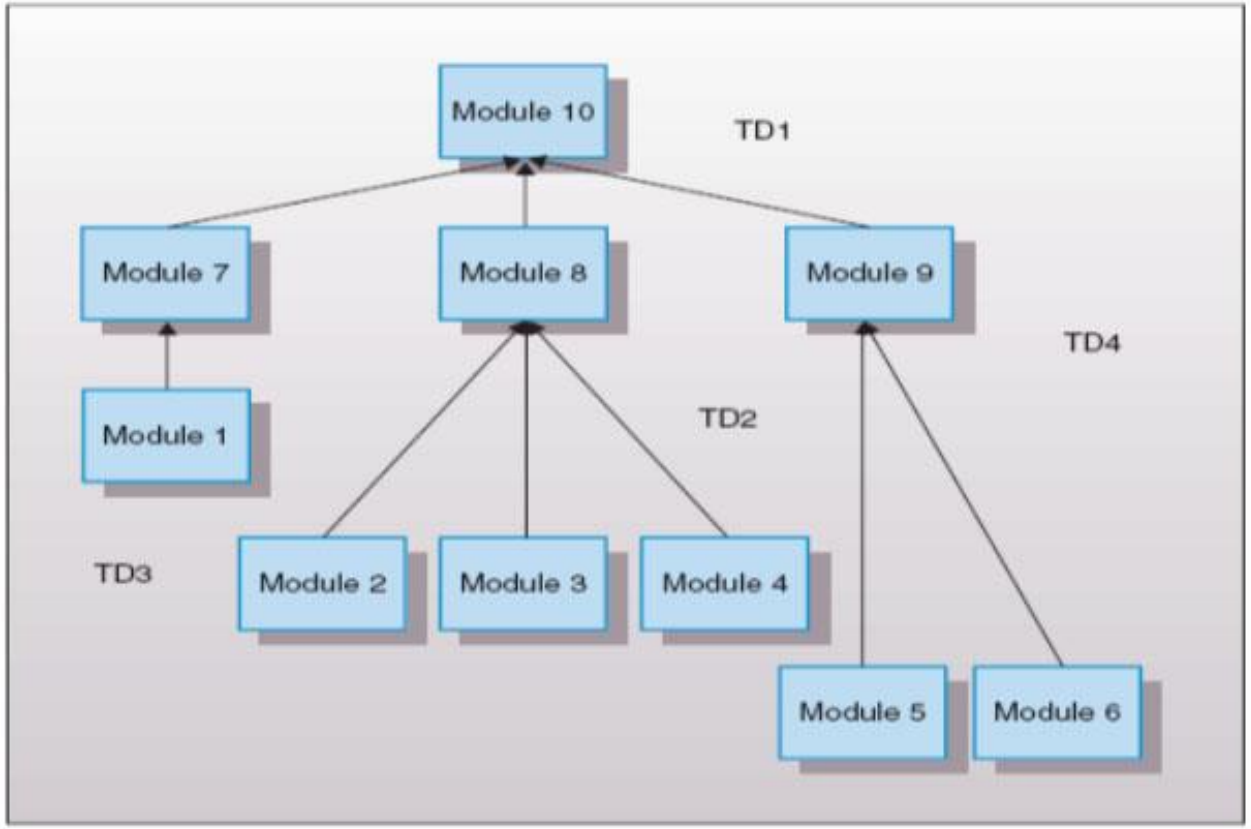
- Modüllerin birleştirilmesi sırasında veri kaybı, dikkatsizlik nedeni ile birbirini ters etkileme, alt fonksiyonların birleştirilmesiyle beklenilen ana fonksiyonunun gerçekleşmemesi, her birinde göze alınabilen hata toleranslarının eklenerek büyümesi, genel veri yapılarının sorun yaratması söz konusudur.
- Bu hata ve sorunları bulup gidermek için, modüllerin birleştirilerek ana programın oluşturulmasında, bütünleme testi uygulanmalıdır.

BÜTÜNLEME (Integration Test) Testi

- Bütünleme testi: a) bütün olarak sınama, b) artırmalı sınama olarak, iki ayrı biçimde birleştirildikten sonra gerçekleştirilmektedir. Bu yüzden, karışıklığa yol açmakta ve sakıncalı bulunmaktadır.
- Artırmalı sınamada, modüller teker teker birbirine bağlanmaktadır. Böylece hatalar kolayca ayrılıp düzeltilebilmektedir. Artırmalı sınama, yukarıdan aşağı ve aşağıdan yukarı olarak iki ayrı şekilde uygulanmaktadır.

Yukarıdan-Aşağı Bütünleme:

- Modüller, ana kontrol modüllünden başlayarak, aşağıya doğru hiyerarşik kontrol yapılarak birleştirilmektedir. Alt basamaklara iniş, bir ana kontrol yolu üzerinden derinliği (depth-first) veya her alt düzeyde yazılara (breadth-first) doğru yürütülmektedir.

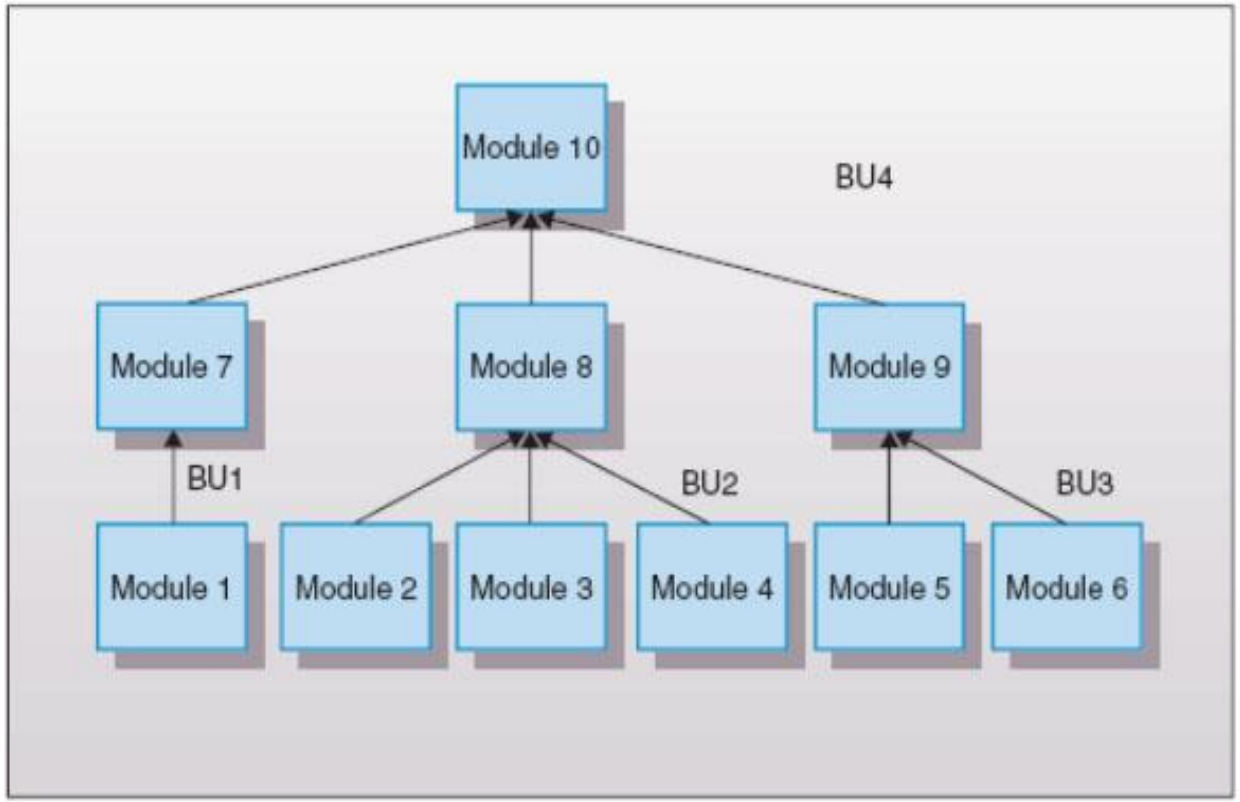


Yukarıdan-Aşağı Bütünleme:

- Yukarıdan aşağıya bütünleme testi, basit ve kullanışlıdır. Ancak, bazı hallerde kodlamanın yukarıdan aşağı tam olarak bağlanması ve birleştirilmesi gerçekleştirilememektedir. Örneğin, alt düzeydeki bir modülü işlemek için, üst düzey girdi verisi bulmak güçlüğü ile karşılaşılmaktadır. 50 – 100 modülden oluşan bir programın her modülün eklenmesinde tekrar çalıştırılması da, önemli bir gideri gerektirmektedir. Bu durumda yöntem, "aşağıdan yukarı bütünleme" ile birlikte uygulanmaktadır (sandwich testing)

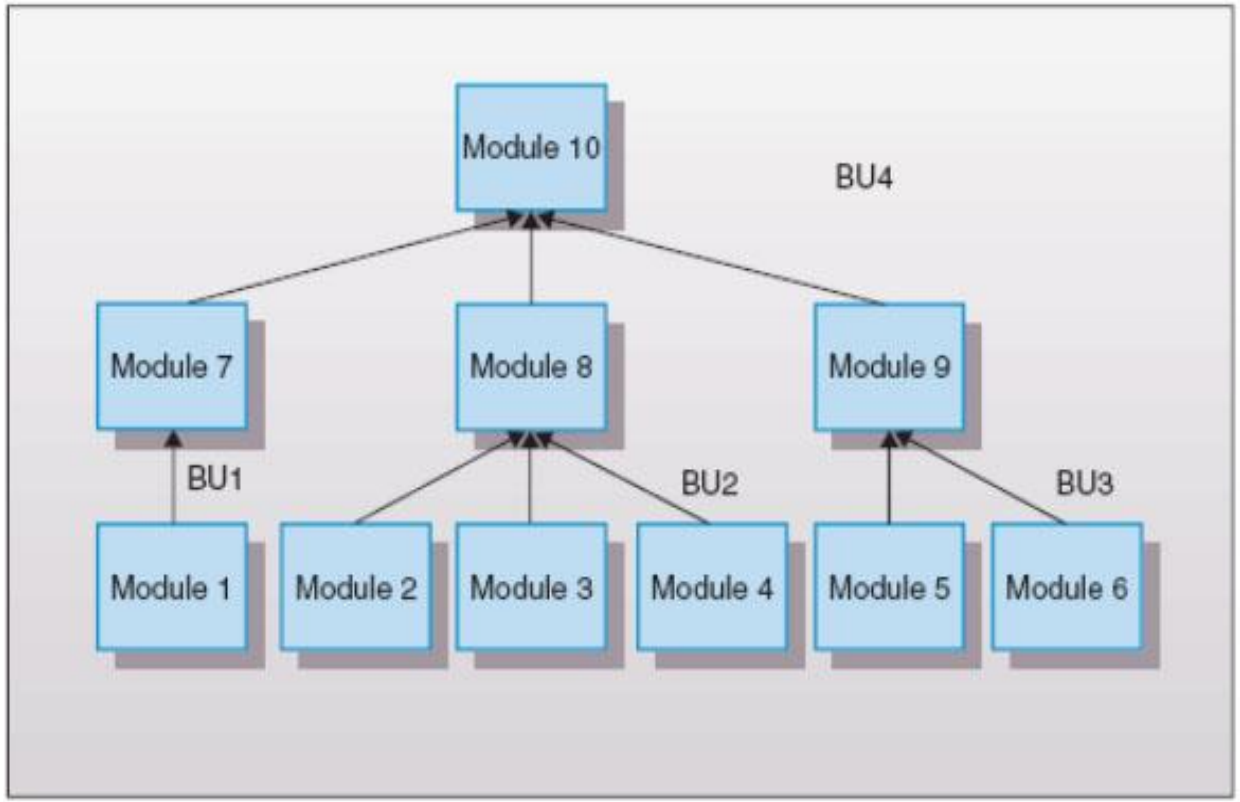
Aşağıdan – Yukarı Bütünleme

- En alt düzeydeki modüllerden (atomic modules) başlayarak, yukarı doğru birleştirilmekte ve sınanmaktadır. Bu yöntemde, alt düzey modül bağlantıları denetlenmiş olduğu için, ayrıca "koçan" oluşturmaya gerek bulunmamaktadır. Yukarıda basamaklara çıkıldıkça, test sürücüleri nin de sayısı azaltılmakta ve böylece kümelerin birleştirilmesi kolaylaşmaktadır.



Aşağıdan – Yukarı Bütünleme

- Bu yöntemin üstünlüğü, “koçan” oluşturmaya gerek bulunmaması ve test programı tasarımının kolay oluşudur. Sakıncası ise, en üst modüle ulaşıncaya kadar programın bütünlük kazanamamasıdır.



Regresyon Testi

- Sınanmış olan bir program veya program parçası üzerinde değişiklik veya ekleme yapılması halinde, tümünün bir kez daha sınanmasıdır.
- Regresyon Testi Uygulama ortamlarında gerekli değişiklikler ve sabitlemeler yapıldıktan sonra yeniden yapılan testlere regresyon testi denilir.
- Başka bir tanımla, Regresyon Testi, önceden test edilmiş bir yazılımın çeşitli değişiklerden geçtikten sonra da hatasız bir şekilde çalışmasını sağlamak amacıyla yeniden test edilmesi işlemidir. Böylece, önceki testlerde belirlenen sorunlar giderildiğinden ve yeni hatalar oluşmadığından emin olunur.

Regresyon Testi

- Regresyon testinin iki amacı vardır;
  1. Sistem kararlılığını sağlamak
  2. Uygun güvenlikteki yazılım kalitesini sağlamak
- Uygulamanın kaç kez yeniden test edilmesi gerektiğini belirlemek güçtür ve bu nedenle, özellikle uygulama geliştirme döneminin sonlarına doğru yapılır. Burada amaç yazılımın değişiklik öncesindeki gibi işlevsel olarak istenilen şekilde çalışmasını sağlamaktır.
- Regresyon testindeki genel yöntemler ise,
  1. önceki testlerin yeniden çalıştırılması ve
  2. önceki sabit hataların yeniden kontrol edilmesi temeline dayanır.

Regresyon Testi

- Nesne Yönelimli Programlamanın kendine has özellikleri regresyon testi sırasında ek işlemlerin yapılmasına neden olur. Örneğin, çok şekillilik (Polymorphism) göz önüne alınarak sistemle ilgili bir dinamik analiz yapılmak istendiğinde kontrol akış diyagramı ve program grafı irdelenmek zorundadır.

SİSTEM TESTİ

- Bilgisayar sistemi, donanım ve yazılım alt sistemlerinden oluşmaktadır. Bu nedenle, yazılım alt sisteminin kendi başına sınanması yeterli olmayıp, bilgisayar sistemi içerisinde de denetlenmelidir.
- Sistem testinin amacı; sistemin bütün öğelerinin uygun olarak bir araya getirildiğinin ve her birinin işlevini tam olarak gerçekleştirebildiğinin onaylanmasıdır. Yazılıma dayalı sistemlerde sistem testi;

1. düzeltme testi,
2. güvenlik testi,
3. dayanıklılık testi,
4. yetenek testi

biçimlerinde uygulanmaktadır.