YMT 312-Yazılım Tasarım Ve Mimarisi

ALT-DÜZEY TASARIM

Bölüm-11

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bu Haftaki Konular

GENEL BAKIŞ

KISIM 1 – Görünürlük, Erişilebilirlik, ve Bilgi Saklama

KISIM 2 – Operasyon, Algoritma ve Veri Yapısı Spesifikasyonu, ve Tasarımın Tamamlanması

KISIM 1

Görünürlük, Erişilebilirlik, ve Bilgi Saklama (Visibility, Accessibility, and Information Hiding)

Amaçlar

- Görünürlük kurallarının program varlıklarının isimleri üzerinden nasıl erişilebilir yaptığını açıklamak
- Referans ve takma-adların görünürlüğün ötesinde nasıl erişilebilirlik sağladığını göstermek
- Bilgi saklama açısından görünürlüğün sınırlandırılmasının ve erişimin genişletilmemesinin önemini vurgulamak
- Erişimin genişletilmesine izin verilebilecek durumları açıklamak

İçerik

- İsimler ve görünürlük
- Referanslar ve takma-adlar
- Erişilebilirlik ve bilgi saklama
- Bilgi saklama için kurallar
- Bilgi saklama pratiklerinin ihlali

Varlıklar, İsimler, ve Görünürlük

- Bir **program varlığı** (program entity) genel olarak program içerisinde bir birim olarak ele alınan herhangi bir şeydir.

- Bir isim (name) bir program varlığıyla bağlantılı bir belirteçtir (identifier).

- Bir program varlığına eğer program içinde herhangi bir noktada ismiyle başvurulabiliyorsa, o varlık o noktada **görünür** (visible) demektir.

Görünürlük Örneği

```
File: package1/PublicClass.java
package package1;
public class PublicClass{
   private String privateAttribute;
   String packageAttribute;
      public void method() {
      String localVariable;
      ...
      // point A
      ...
   }
   ...
   // point B
   ...
} // end package1.PublicClass

File: package1/PackageClass.java
package package1;
class PackageClass{
   ...
   // point C
   ...
} // end package1.PackageClass

File: package2/PackageClass.java
package package2;
import package1.*;
class PackageClass{
   ...
   // point D
   ...
} // end package2.PackageClass
```

Görünürlük Türleri

- **Yerel** (Local)—Yalnızca tanımlandığı modül içerisinde görünürdür.
- **Yerel-olmayan** (Non-local)—Tanımlandığı modül dışında da görünürdür, ancak programın her yerinde değil.
- **Global (Global)**—Programın her yerinde görünürdür.
- Eğer bir varlık tanımlandığı modül dışında da görünür durumda ise o varlık tanımlandığı modülden **ihraç edilmiş** (exported) demektir.

Nesne Yönelimli Nitelik Görünürlüğü

Private—Yalnızca tanımlandığı sınıf içerisinde görünürdür

- Yerel görünürlüğün bir türü

Package—Yalnızca tanımlandığı sınıf ve bu sınıfla birlikte aynı paket (veya namespace) içerisinde bulunan sınıfların içerisinde görünürdür

- Yerel-olmayan görünürlüğün bir türü

Protected—Yalnızca tanımlandığı sınıf ve bu sınıftan türeyen alt-sınıflar içerisinde görünürdür

- Yerel-olmayan görünürlüğün bir türü

Public—Sınıfın görünür olduğu her yerde görünürdür

- Yerel-olmayan ve global görünürlüklerin bir türü

Erişilebilirlik (Accessibility)

Bir program varlığı eğer program metni içinde herhangi bir noktada kullanılabiliyorsa o noktada **erişilebilir** (accessible) demektir.

Bir program varlığı görünür olduğu yer yerde aynı zamanda erişilebilirdir.

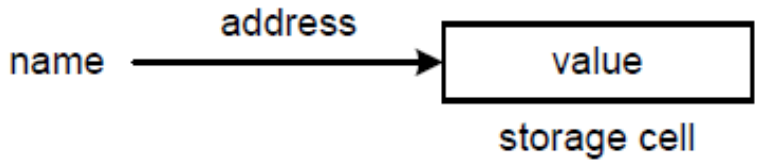
Bir program varlığı görünür olmadığı yerlerde de erişilebilir olabilir.

Değişkenler (Variables)

Bir **değişken** (variable) değerleri saklamak için kullanılan bir programlama aygıtıdır.

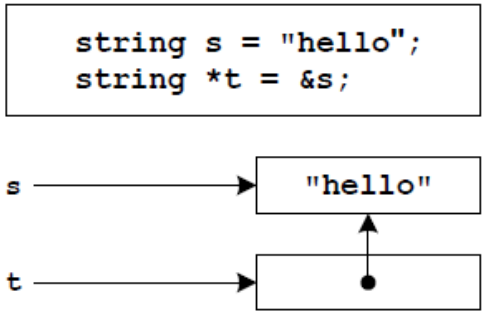
Değişkenlerin nitelikleri:

- İsim (Name)
- Değer (Value)
- Adres (Address)



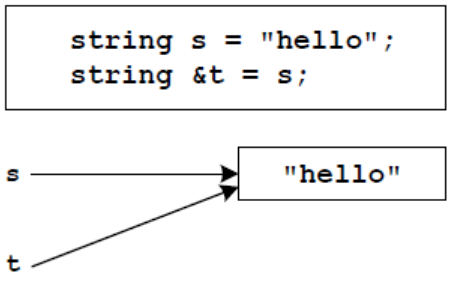
Referanslar (References)

Bir **referans** (reference) bir değerin saklandığı bellek adresini içeren bir ifadedir.



Takma-adlar (Aliases)

Bir **takma-ad** (alias) başka bir değişkenle aynı adrese sahip bir değişkendir.



Görünürlüğün Ötesinde Erişilebilirlik

Referanslar ve takma-adlar değişkenleri görünür olmadıkları yerlerde de erişilebilir yapabilirler

- Parametre olarak bir referans geçirilmesi
- Bir parametrenin referans ile geçirilmesi (takma-ad ile)
- Bir alt-programdan bir referans döndürülmesi

Bu kullanım şekline **erişilebilirliğin görünürlüğün ötesine genişletilmesi** (extending access beyond visibility) denir.

- Genellikle iyi bir programlama pratiği değildir (bad practice)

Bilgi Saklama ve Erişim

Bilgi saklamada anahtar teknik program varlıklarına erişimi mümkün olduğunca kısıtlamaktır.

- **Görünürlüğü kısıtlamak** (Limiting visibility)—Kapsam ve görünürlük işaretleri kullanarak görünürlüğü sınırlandırmak
- Erişimi genişletmemek (Not extending access)—Görünürlüğü genişletmek için referanslar ve takma-adlar kullanmaktan sakınmak

Defansif kopya (defensive copy) tekniği, bir operasyondan normalde görünür olmayan bir nesnenin referansı yerine nesnenin tam bir kopyasının referansının döndürülmesi taktiğidir.

Bilgi Saklama Kuralları 1

Görünürlüğü sınırlandırın.

- Program varlıklarını yalnızca mümkün olan en küçük program bölgesinde görünür yapın.
- Deklarasyonların kapsamını mümkün olan en küçük program bölgesiyle sınırlı tutun.
- **Özellikleri (attribute) en azından protected, tercihan da private yapın.**
- Yardımcı operasyonları en azından protected, tercihan da private yapın
- **Global görünürlükten uzak durun.**
- **Paket görünürlüğünden uzak durun.**

Bilgi Saklama Kuralları 2

Erişimi genişletmeyin.

- Özellikleri sınıfa parametre geçirilen referanslarla ilişkilendirmeyin—bunun yerine defansif kopyalarını oluşturun.
- Özelliklere referans durumunda olan değişkenleri parametre olarak geçirmeyin veya geri döndürmeyin—bunun yerine defansif kopyalarını geçirin veya döndürün.
- Parametreleri referansla geçirmeyin.
- Takma-adlar kullanmayın.

Erişimin Genişletilmesinde İstisnalar

Şu iki durumda erişimi görünürlüğün ötesine genişletmek gerekebilir:

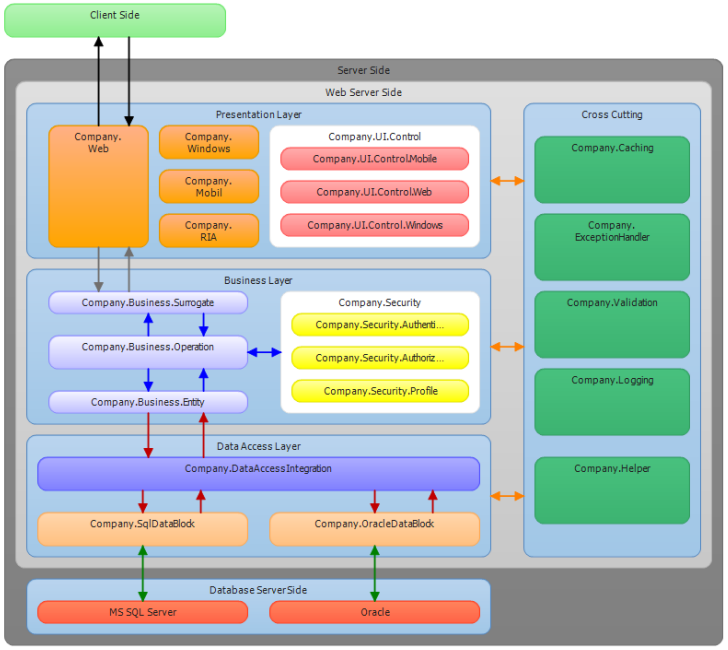
- Modüller işbirliği için bir varlığı paylaşmak zorunda olabilir
  - Örneğin: bir paylaşılan kuyruk (a shared queue)
- Birtakım tasarım hedefleri bilgi saklamadan daha yüksek öneme sahip olabilir
  - Örneğin: performans kısıtları

Özet

- Program varlıkları genellikle programın çeşitli bölgelerinde görünür olarak isimleriyle erişilirler.
- Varlıklar ayrıca referanslar ve takma-adlarla da erişilebilirler.
- Bilgi saklama prensibine göre görünürlük kısıtlı olmalı ve erişim görünürlüğün ötesine genişletilmemelidir.
- Bazı özel durumlarda bu prensip ihlal edilebilir.

KISIM 2

Operasyon, Algoritma ve Veri Yapısı Spesifikasyonu, ve Tasarımın Tamamlanması



Amaçlar

Operasyon spesifikasyonlarını ve bunların içeriklerini sunmak

Operasyon davranışlarının bildirimsel spesifikasyonu için sözleşme ile tasarım kavramını sunmak

Algoritma spesifikasyonu için minispecs ve sözde-kodu (pseudocode) tanıtmak

Veri yapısı spesifikasyonu için veri yapısı diyagramlarını tanıtmak

Tasarım sonlandırmayı araştırmak

İçerik

- Operasyon spesifikasyonları
- Bildirimsel ve prosedürel davranış spesifikasyonu
- Sözleşme ile tasarım (Design by contract)
- Assertion, precondition, postcondition, ve class invariant
- Algoritma spesifikasyonu
- Veri yapısı spesifikasyonu
- Tasarımın tamamlanması

Operasyon Spesifikasyonu (Op-Spec)

Operasyonun arabirimini ve sorumluluklarını gösteren yapısal metin

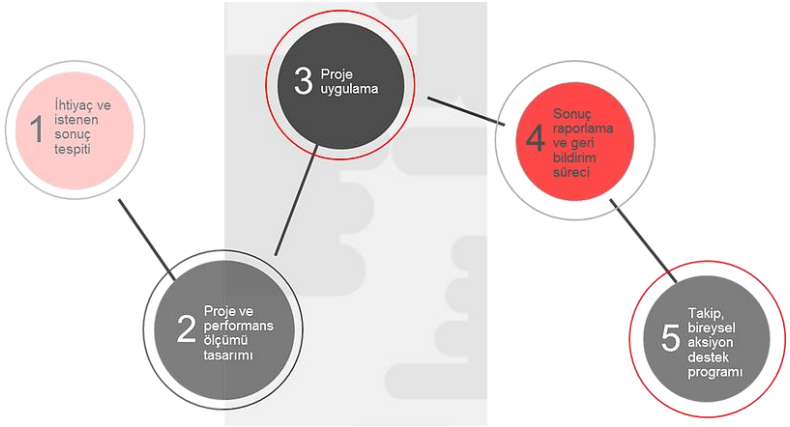
- **Sınıf veya modül** (Class or module)—Operasyonu belirtir
- **İmza** (Signature)—Operasyonun adı, parametrelerin adları ve tipleri, dönüş tipi, ve belki daha fazlası (syntax)
- **Açıklama** (Description)—Bir iki cümle ile
- **Davranış** (Behavior)—Semantikler ve pragmatikler
- **İmplementasyon** (Implementation)—İsteğe bağlı

Davranış Spesifikasyonu

Prosedürel (Procedural)—Girdileri çıktılara dönüştüren bir algoritmayı açıklar

- Bir **algoritma** (algorithm) bir bilgisayar tarafından gerçekleştirilebilen işlem adımları sırasıdır.

Bildirimsel (Declarative)—Algoritma belirtmeden girdileri, çıktıları, çağırma kısıtlarını, ve sonuçlarını açıklar



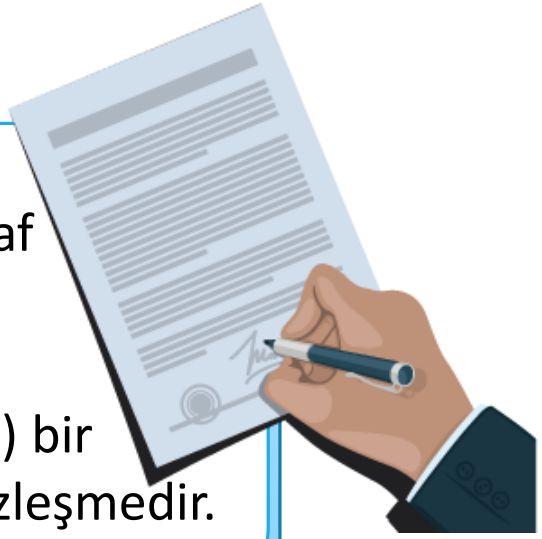
Bildirimsel Spesifikasyonun Avantajları

- Daha soyuttur çünkü implementasyon ayrıntılarını gözardı eder—daha öz/özet
- Arabirime odakların, içsel işleyişe değil
- Prosedürel spesifikasyonun yaptığı gibi programcıyı belirli bir implementasyonun uygulanması için yönlendirmez, zorlamaz.

Sözleşme ile Tasarım (Design by Contract)

Bir **sözleşme** (contract) iki veya daha fazla taraf arasında bağlayıcı bir anlaşmadır.

Bir **operasyon sözleşmesi** (operation contract) bir operasyon ve onu çağıranlar arasındaki bir sözleşmedir.



Sözleşme Hakları ve Yükümlülükleri

- Çağıran
  - Uygun şartlar altında geçerli (valid) parametreler geçirmekle yükümlüdür.
  - Duyurulan işlemsel hizmetleri alma hakkına sahiptir.

- Çağrılan operasyon
  - Duyurulan hizmetleri sağlamakla yükümlüdür.
  - Uygun koşullar altında geçerli parametrelerle çağrılma hakkına sahiptir.

Bildirimler (Assertions)

Bir **assertion** bir program içinde belirlenen noktada doğru (true) olması gereken bir ifadedir

- Bu bildirimler çağıran ve çağrılan operasyonun hakları ve yükümlülüklerini belirtir.

Önkoşul ve Sonkoşullar

Bir **önkoşul** (precondition) bir operasyonun en başında doğru olması gereken bir assertion'dır.

Bir **sonkoşul** (postcondition) bir operasyon tamamlandığında doğru olması gereken bir assertion'dır.

- Önkoşullar çağıranın yükümlülüklerini ve çağrılan operasyonun haklarını belirtirler.
- Sonkoşullar ise çağıranın haklarını ve çağrılan operasyonun yükümlülüklerini belirtirler.

Operasyon Spesifikasyonu Örneği

Signature	public static int findMax(int[] a) throws IllegalArgumentException
Class	Utility
Description	Return one of the largest elements in an int array.
Behavior	pre: (a != null) && (0 < a.length) post: for every element x of a, x <= result post: throws IllegalArgumentException if preconditions are violated

Sınıf Değişmezleri (Class Invariants)

Bir **önkoşul** (precondition) bir operasyonun en başında doğru olması gereken bir assertion'dır.

Bir **sonkoşul** (postcondition) bir operasyon tamamlandığında doğru olması gereken bir assertion'dır.

- Sınıf değişmezleri bütün ihraç edilen operasyonların sözleşmelerini tamamlayıcı niteliktedir.

Bildirimlerde (Assertion) Neler Olmalı 1

Önkoşullar (Preconditions):

- Parametreler üzerindeki kısıtlamalar
- Operasyon çağrılmadan önce sağlanması gereken koşullar

Sonkoşullar (Postconditions):

- Parametreler ve sonuçlar arasındaki ilişkiler
- Sonuçlar üzerindeki kısıtlamalar
- Parametrelerdeki değişiklikler
- İhlal edilen önkoşullara karşı verilecek karşılıklar

Bildirimlerde (Assertion) Neler Olmalı 2

Sınıf değişmezleri (Class invariants)

- Özellikler (attribute) üzerindeki kısıtlamalar
- Özellikler (attribute) arasındaki ilişkiler

Boş bildirimleri “true” veya “none” olarak ifade edin.

Op-Spec'lerin Oluşturulması

Orta-düzey tasarımın ilk evrelerinde ayrıntılı op-spec'ler oluşturmayın

- Çünkü tasarım henüz çok değişken; ayrıntılar çok fazla değişikliğe uğrayabilir

Tasarımın sonunu beklemeyin

- Ayrıntılar unutulabilir
- Muhtemelen çok yetersiz kalır

Op-spec'leri tasarım sırasında kademeli olarak oluşturun, kesinleştikçe de ayrıntıları ekleyin

Algoritma Spesifikasyonu

- İyi-bilinen algoritmaları yalnızca adıyla belirtin.
- **Minispec** kullanarak bir algoritmanın, girdileri çıktılara nasıl dönüştürdüğünü adım adım anlatın.
- Minispec'leri **sözde kod** (pseudocode) ile yazın.

Pseudocode Örneği

```
Inputs: array a, lower bound lb, upper bound ub,
    search key
Outputs: location of key, or -1 if key is not
    found
lo = lb
hi = ub
while lo <= hi and key not found
   mid = (lo + hi) / 2
   if      ( key = a[mid] ) then key is found
   else if ( key < a[mid] ) then hi = mid-1
   else                          lo = mid+1
if key is found then return mid
else                 return -1
```

Veri Yapıları

Bir **önkoşul** (precondition) bir operasyonun en başında doğru olması gereken bir assertion'dır.

Bir **sonkoşul** (postcondition) bir operasyon tamamlandığında doğru olması gereken bir assertion'dır.

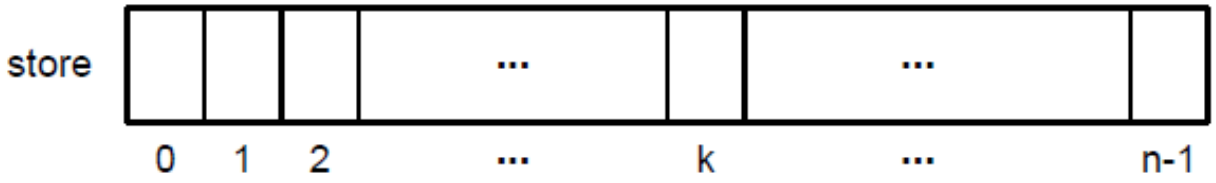
Ardışık (Contiguous) implementasyon—Değerler birbirine bitişik bellek hücrelerinde saklanır

Bağlı (Linked) implementasyon—Değerler referanslar veya işaretçilerle erişilebilen rastgele bellek hücrelerinde saklanır

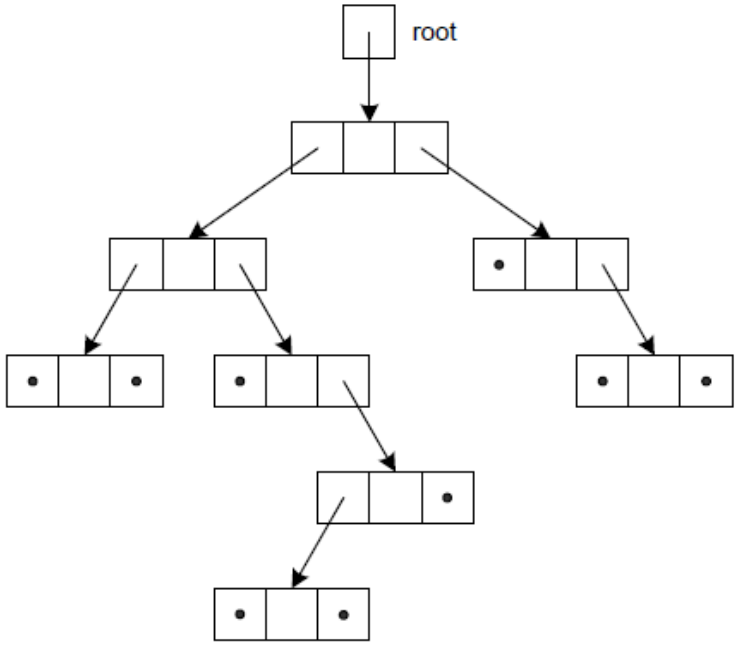
Veri Yapısı Diyagramı

- Dikdörtgenler bellek hücrelerini temsil eder. Genellikle isimlendirilirler.
- Bitişik hücreler bitişik dikdörtgenlerle temsil edilir. Hücrelerin indisleri olabilir.
- Tekrar eden elemanlar ... (ellipsis) ile gösterilir.
- Bağlı hücreler işaretçileri veya referansları temsil eden, bir hücreden diğerine giden oklarla gösterilir.
- Null işaretçiyi göstermek için nokta kullanılır.

Veri Yapısı Diyagramı Örneği 1

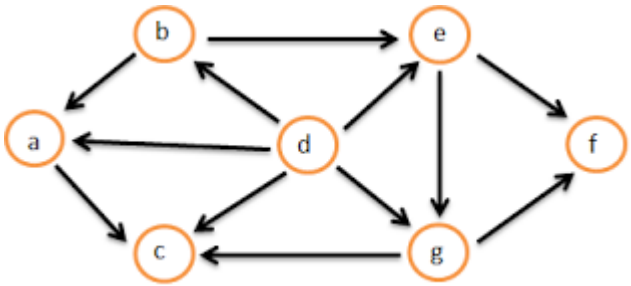


Veri Yapısı Diyagramı Örneği 2



Veri Yapısı Diyagramı Kuralları

- Record alanlarını yalnızca bir kez etiketleyin.
- Büyük ve tekrar eden yapıları ellipsis (…) ile basitleştirin.
- Bağlı yapıları çizerken işaretçilerin okları yukarıdan aşağıya veya soldan sağa doğru ilerlesin.
- Az bilinen veya ilave semboller için lejant (gösterge) kullanın.



Tasarımın Tamamlanması

- Alt-düzey spesifikasyonlarla birlikte tasarım dokümanı tamamlanır.
- Tasarımın tamamlanması (Design finalization) aşamasında tasarımın kalitesi ve iyi dokümante edildiğinden emin olmak için kontroller yapılır.
- Mühendislik tasarımı sürecinin son adımıdır.

Tasarım Dokümanı Kalite Karakteristikleri 1

Fizibilite (Feasibility)—Tasarım gerçeklenebilir, yapılabilir olmalıdır

Yeterlilik (Adequacy)—Tasarım gereksinimleri karşılayacak bir programı belirtmelidir

Ekonomi (Economy)—Tasarım zamanında ve bütçe sınırları içinde yapılabilecek bir programı belirtmelidir

Değişebilirlik (Changeability)—Tasarım kolayca değiştirilebilecek bir programı belirtmelidir.

Tasarım Dokümanı Kalite Karakteristikleri 2

İyi yapılanmışlık (Well-Formedness)—Tasarımda notasyonlar doğru kullanılmalıdır

Tamlık (Completeness)—Programcıların programı yazabilmeleri için gereken herşeyi belirtmelidir

Açıklık/Netlik (Clarity)—Mümkün olduğunca anlaşılması kolay olmalıdır

Tutarlılık (Consistency)—Tek bir ürün tarafından gerçekleştirilebilecek bir tasarım içermelidir.

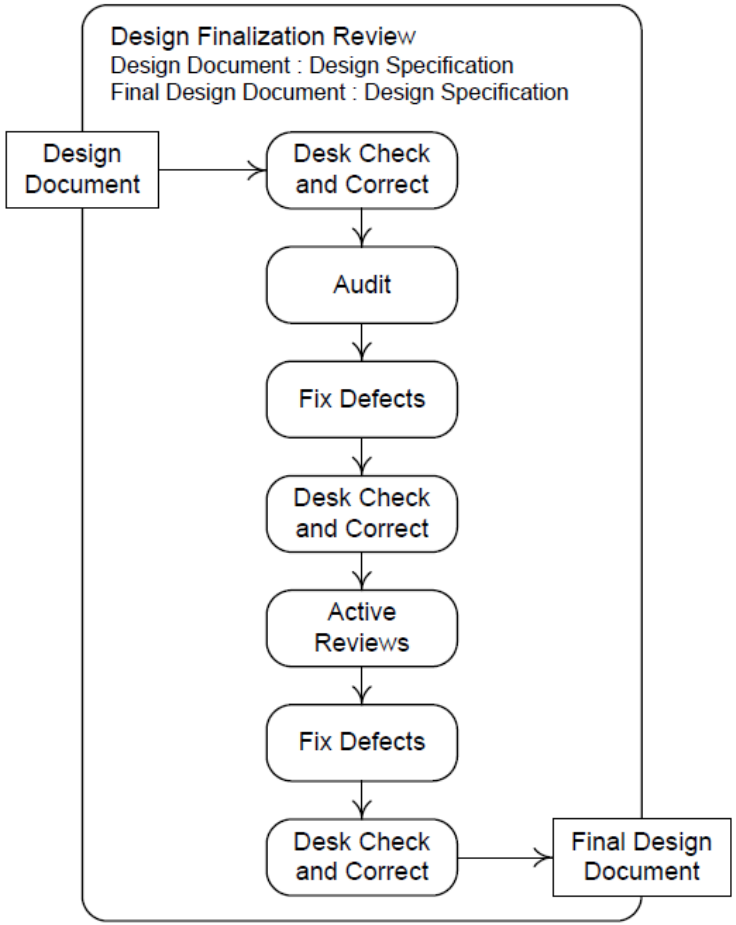
Eleştirel Gözden Geçirme (Critical Review)

Bir **eleştirel gözden geçirme** (critical review) tamamlanmış bir ürünün kabul edilebilir bir kalitede olup olmadığına dair değerlendirilmesidir.

Gözden geçirmeler şu şekillerde olabilir:

- Desk checks,
- Walkthroughs,
- Inspections,
- Audits, and
- Active reviews.

Eleştirel Gözden Geçirme Süreci



Sürekli (Continuous) Gözden Geçirme

- Eleştirel gözden geçirme sayesinde tasarımın erken safhalarına dönülmesini gerektirecek çok ciddi tasarım kusurları bulunabilir. Bu da;
  - Pahalıdır
  - Zaman kaybettiricidir
  - Sinir bozucudur

- Sürekli gözden geçirme (continuous review) politikası sayesinde tasarımın her aşamasında kusurlar daha erken fark edilebilir ve geç fark edildiğinde yaşanabilecek sıkıntıların önüne geçilebilir.

Özet

- Operasyon spesifikasyonları operasyonlarla ilgili tasarım ayrıntılarını belirtiler
  - Sınıf veya modül adı
  - İmzası
  - Açıklaması
  - Davranışı
  - İmplementasyonu
- Davranışlar bildirimsel veya prosedürel olarak belirtilebilir.
- Bildirimsel spesifikasyon assertion'lar ile belirtilen operasyon sözleşmeleriyle yapılır.
  - Önkoşullar
  - Sonkoşullar

Özet

- Algoritmalar minispec veya pseudocode ile belirtililebilirler.
- Veri yapıları veri yapısı diyagramı kullanılarak gösterilirler.
- Tasarımın tamamlanması, mühendislik tasarımının son adımıdır.
- Tasarım dokümanı kalite karakteristiklerini karşılaması bakımından bir eleştirel gözden geçirme süreci ile kontrol edilir.
- Eleştirel gözden geçirmelerin kurum politikası olarak sürekli gözden geçirme şeklinde uygulanması geç fark edildiğinde felakete yol aşabilecek kusurların erken safhalarda fark edilmesini sağlar.

Kaynaklar

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş”, TBİL-211, Dr. Ali Arifoğlu.

”Yazılım Mühendisliği” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)

Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

http://www.cclub.metu.edu.tr/bergi_yeni/e-bergi/2008/Ekim/Cevik-Modelleme-ve-Cevik-Yazilim-Gelistirme

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2011/ch6_6d_sk

http://dsdmofagilemethodology.wikidot.com/

http://caglarkaya.piquestion.com/2014/07/01/244/