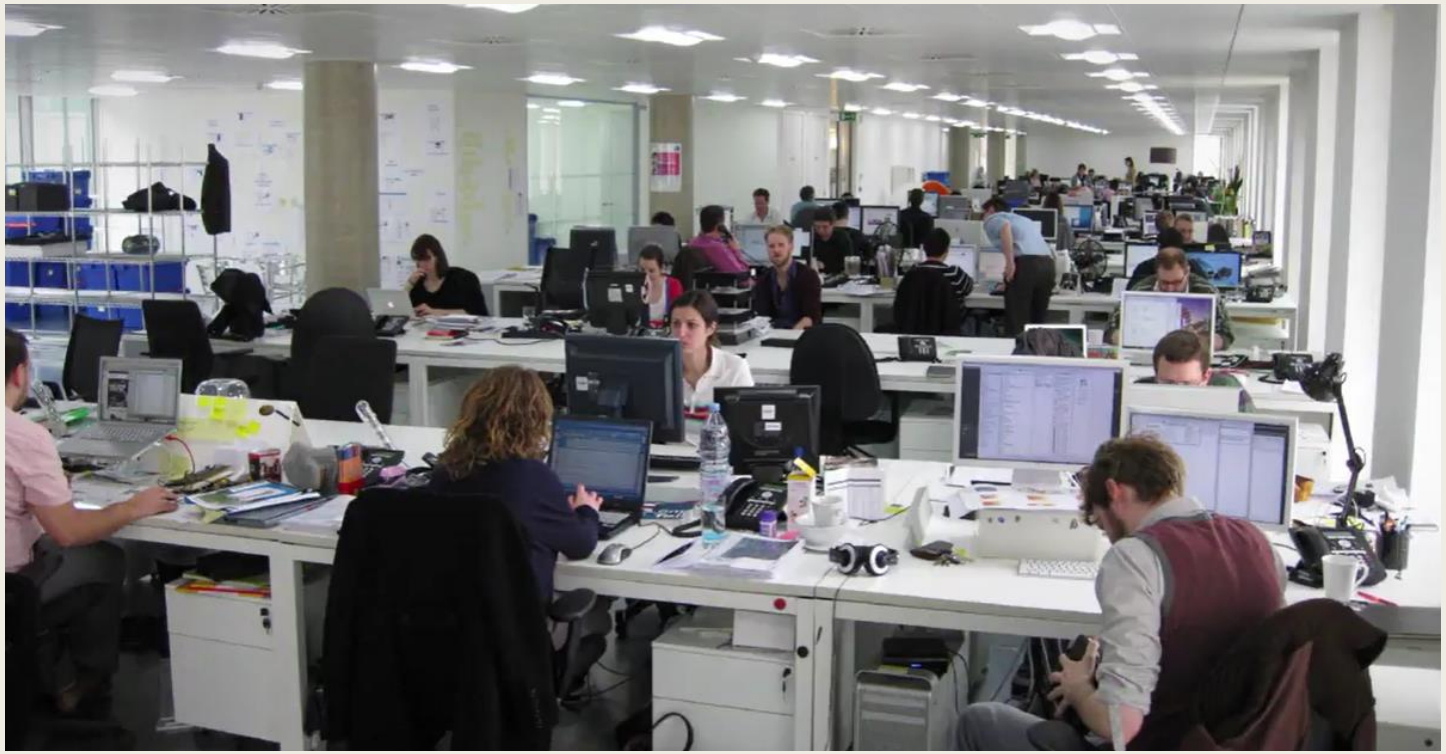
BILGISAYAR MUHENDISLIĞI TASARIM

Dr.Öğr.Üyesi Betül AY

HAFTA-2- Yazılım Sürecinde Gereksinim ve Tasarım

Yazılım Tasarımı ve Mimarisinin Endüstirideki Rolü

- Yazılım tasarımı ve mimarisinde bir kariyer düşünmek neye benzemektedir?
- Yazılım tasarımı ve yazılım mimarisi arasındaki fark nedir?



Yazılım Tasarımı ve Mimarisi

- Yazılım tasarımı, bir müşterinin isteklerini ve gereksinimlerini uzun vadede istikrarlı ve sürdürülebilir olan, geliştirilebilen ve daha büyük bir sistemin parçası olabilecek çalışma koduna dönüştürme işlemidir.
- Bir yazılım mimarının görevi, ürün ile müşteri ve mühendis ekipleri arasındaki ara yüz olmaktır .
- Yazılım mimari, müşterilerin yazılım gereksinimlerine ve taleplerine göre teknik gereksinimleri ortaya çıkarmaktır. Temel amacı, müşterinin ihtiyacına, müşterinin sahip olduğu bütçe dahilinde hizmet etmektir. Tüm sisteme bakmaktan ve uygun platformların seçilmesinden, veri depolamasından, çözümlerden ve bileşenlerin birbirleriyle nasıl etkileşime gireceğinin belirlenmesinden sorumlu olacaktır.

Yazılım Sürecinde Tasarım

- Yazılım geliştirildiğinde, genellikle bir süreçten (**process**) geçer.
- Basit bir ifadeyle, bir process bir problem alır ve yazılımı içeren bir çözüm yaratır.
- Bir process yinelemelidir (**iterative**).
- Bu iterasyonlar tespit edilen problemlere dayalı bir grup gereksinimden oluşmaktadır ve bu gereksinimleri kavramsal tasarım modelleri (**conceptual design mock-ups**) ve teknik tasarım diyagramları (**technical design diagrams**) oluşturmak için kullanmaktadır.
- Bu process her bir ihtiyaç grubu için tekrarlanır ve sonuçta proje için eksiksiz bir çözüm oluşturulur.
- Bu process atlandığında, özellikle iş derhal kodlama ile başladığında ve gereksinimlerin ve tasarımın anlaşılmaması durumunda birçok proje başarısız olur ☹

Yazılım Sürecinde Tasarım

Standish Grubu (https://www.standishgroup.com/) tarafından yapılan bir ankette, ankete katılanların% 13'ü eksik gereksinimlerin projelerini bozduğunu belirtti! Doğrudan uygulama çalışmasına dalmak, projenin başarısız olmasının önde gelen bir nedenidir.

Yazılım Sürecinde Gereksinim ve Tasarım

Gereksinim (Requirements)

- Gereksinimler, müşteri (client) veya kullanıcı isteğine (user request) bağlı olarak bir üründe uygulanması gereken şartlar veya yeteneklerdir.
- Bunlar bir projenin başlangıç noktasıdır, "müşterinizin ne istediğini anlamalısınız".
- Gereksinimleri ortaya çıkarmak için, müşterinin vizyonundan daha fazlasını sormak ve istemek önemlidir:
  - *Müşteri vizyonunun araştırılması*
  - *Neyi eksik söylediğinin ortaya çıkarılması*
  - *Müşterinin göz önünde bulundurmadığı sorunlar hakkında sorular sorulması…*
- Kodlamaya başlamadan önce ne ürettiğinizi ve müşterinizin bir üründen ne istediğini tam olarak anlamanız gereklidir.

Yazılım Sürecinde Gereksinim ve Tasarım

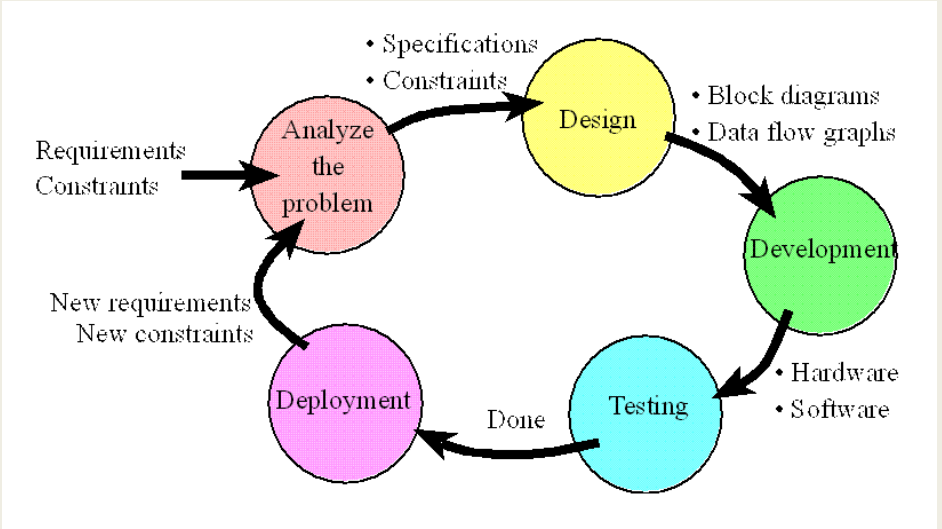
Tasarım (Design)

- Gereksinimler oluşturulduktan sonra bir sonraki adım kavramsal ve teknik tasarımdır.
- Kavramsal tasarım bir ürünün nasıl çalışacağını göstermek ve tartışmak için basit bir yol sağlayarak müşteriler ve kullanıcılar ile tasarım kararlarını netleştirmeye yardımcı olur.
- Teknik tasarımlar, çözümün teknik ayrıntılarını tanımlamak için kavramsal tasarımlar ve gereksinimler üzerine kuruludur.
- Kavramsal tasarımda, geliştirilmekte olan yazılımın ana bileşenleri ve bağlantıları ile bunların ilişkili sorumlulukları ana hatlarıyla belirtilmiştir.
- Teknik tasarım bu bilgiyi bir sonraki aşamaya getirir - bu sorumlulukların nasıl yerine getirildiğini açıklamayı amaçlar.

Yazılım Sürecinde Gereksinim ve Tasarım

- Sistem geliştirme sürecinin bir life cycle olarak ele alın.
- Gereksinimleri alın ve problemleri analiz edin.
- Çözümü tanımlamak için bir algoritma oluşturun.
- Algoritmayı sözde kod(pseudo code) ya da akış diyagramı ile gösterin.
- Oluşturduğunuz algoritmayı koda çevirin.
- Kodunuzu test edin.

(Test—>Kod Yaz—>Test—>Kod Yaz…)



http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C7_DesignDevelopment.htm

Kalite Nitelikleri için Tasarım

- Yazılım geliştirilirken istenilen gereksinimlerin nasıl elde edileceği üzerine ayrıntılara yoğunlaşmak önemlidir. Bu nedenle bu derste yazılım oluşturmada gerekliliklerin ve tasarımın önemine dikkat çekilecektir.
- Tasarım konusunda uzlaşma gerektiren bazı kısıtlamalar (restrictions) vardır.
- İstenilen işlevselliğe dayalı yazılım gereksinimlerinin yanı sıra, bu işlevselliğin ne kadar iyi çalışması gerektiğini tanımlayan kalite özellikleri de vardır.
- Örneğin, bir evin ön kapısını tasarlamayı düşünelim. Güvenlik, önemli olabilecek kalite niteliğidir, ancak kapıya çok fazla kilit eklerseniz, kolayca açılması zor olabilir ve kullanımı zor olabilir. İyi bir tasarım güvenliği kolaylık ve performansla dengelemelidir.

Bağlam ve Sonuçlar (Context and Consequences)

- **Bağlam**, tasarımdaki kalitenin dengesine karar verirken önemli bilgiler sağlar.

Örneğin, halkın erişebileceği kişisel bilgileri depolayan yazılımlar, yalnızca şirket çalışanları tarafından kullanılan yazılımlardan farklı güvenlik gereksinimlerine sahip olacaktır.

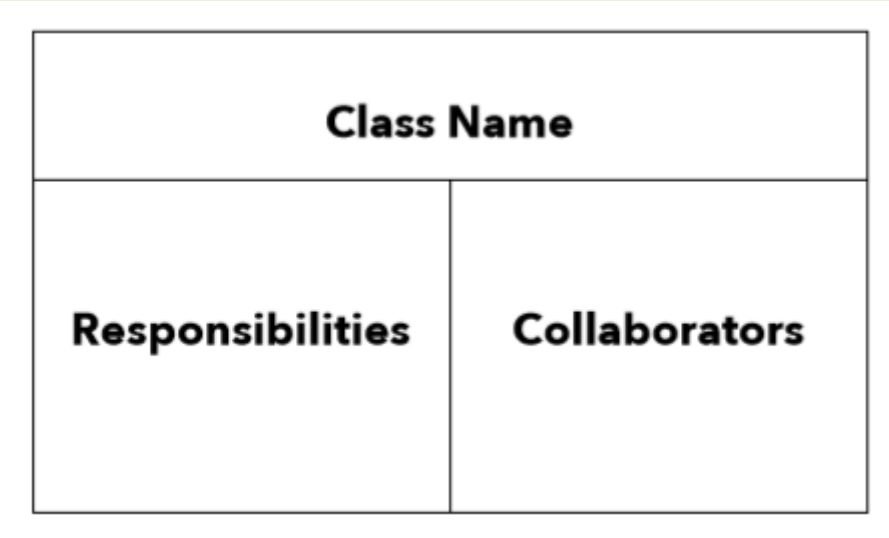
- Yazılım tasarımı **sonuçları** da dikkate almalıdır. Bazen, yazılım tasarımında yapılan seçimlerin istenmeyen sonuçları olabilir.

Örneğin, az miktarda veri için iyi işleyen bir fikir, büyük miktarda veri için pratik olmayabilir.

- Nitelikler arasındaki denge (balance), tasarım sırasında anlaşılmalı ve dikkate alınmalıdır. Hangi niteliklerin gerekli olduğunu öncelik sırasına koymak ve anlamak önemlidir.
- Ürünü geliştirmek için kalitelerin maliyet, zaman ve insan gücü gibi kaynaklarla dengelenmesi gerekir.

Sınıf Sorumluluk İşbirliği (Class Responsibility Collaborator)

Kavramsal tasarımı oluştururken bileşenleri (components), sorumlulukları (responsibilities) ve bağlantıları (connections) yüksek düzeyde temsil etmeye yardımcı olacak önemli bir teknik vardır. Bu teknik, Class, Responsibility, Collaborator (CRC) kartlarının kullanımıdır. CRC kartları üç bölümden oluşur, sınıf adı, sınıfın sorumlulukları ve ortak çalışanlar.



ÖRNEK CRC

- CRC kartlarının birlikte nasıl çalıştığını daha iyi anlamak için, farklı kurslara kayıtlı öğrencileri modellememiz gereken basit bir öğrenci kayıt problemini düşünelim.
- Öğrenci CRC Kartını, öğrencinin adı, öğrenci kimliği ve öğrencinin bir derse kaydolmasını veya dersi bırakmasını sağlayan sorumluluklar gibi özellikleri kolayca tanımlayabiliriz.
- Bu örnekte, işbirlikçi sınıf “ders” sınıfı olacaktır.
- Ders CRC Kartı, ders kodu, ders adı gibi soumlulukları ve “eğitmen” gibi işbirlikçi sınıfından oluşacaktır.

Öğrenci ve Ders CRC Kartı

Öğrenci	
Öğrenci İsmi Öğrenci No Bölümü Ders Kayıt Ders Çıkarma	Ders Sınıfı

Ders	
Ders İsmi Ders Kodu Ders İçeriği Ders Eğitmeni	Eğitmen Sınıfı

CRC Kartlarının Avantajları

- Karmaşık bir sistemi kolayca anlamamızı sağlar. Temelde, işbirlikçi sınıflar arasındaki etkileşimleri birer birer yavaş yavaş inşa ederek daha karmaşık tasarımlar oluşturmaya izin verir.
- CRC kartları basit bir tekniktir ve çok az eğitim almış herkes tarafından kolayca kullanılabilir ve pahalı bilgi işlem kaynakları gerektirmez (bir tahta veya bir kağıt ve bir kalem yeterli olacaktır).
- CRC temel olarak, bir takımdaki farklı kişilerin birlikte çalışarak tasarımı geliştirmelerini ve takımdaki herkesin katkıda bulunmalarını sağlayan bir beyin fırtınası aracıdır.
- CRC, Extreme Programming gibi diğer resmi nesne yönelimli tasarım metodolojileriyle kullanılabilir ve Unified Modeling Language (UML) gibi modelleme dilleriyle birlikte kullanılabilir.