BİM539,
Ders 3:
Gereksinimler

Gereksinim nedir?

- Yazılımın özellikleridir
  - Yazılım Gereksinim Belirtimi (Software Requirements Specification, SRS) adı verilen bir dosyada belirtilir
  - SRS yüzlerce sayfa uzunluğundadır
- Bitmiş bir yazılımın bütün gereksinimleri karşılaması gerekir
- Geliştiriciler bu şekilde neyi kodlayacağını bilirken yazılım testi geliştiricileri de neyi test edeceklerini bilirler.

Gereksinimlere ihtiyacınız var mı ?

- Sayfalarca uzunlukta belge yığınına ihtiyacınız var mı?
- Evet?
- Hayır?

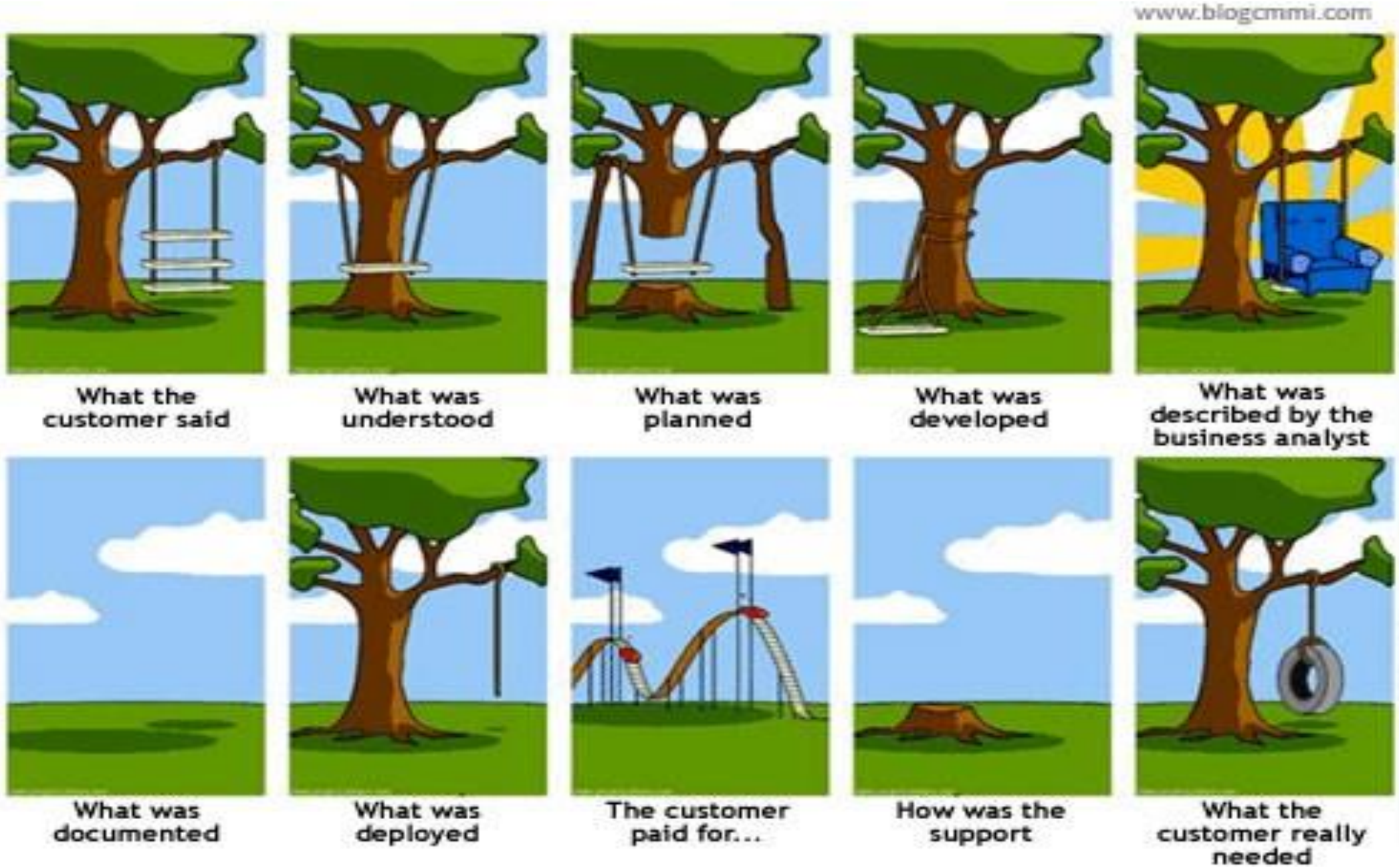
Gereksinim Örneği – Kuş Kafesi

- Kafes 120 cm uzunlukta olacaktır.
- Kafes 200 cm genişlikte olacaktır.
- Kafes paslanmaz çelikten yapılacaktır.
- Kafeste küçük bir kuş için uygun büyüklükte 1 yemek ve 1 su kabı bulunacaktır.
- Kafeste 2 adet tünek bulunacaktır.
- Kuşların en az %90'ı kafesi beğenecektir.

Gereksinimlerimiz ile ilgili problemler?

- Kafes uzunluğu 120.001 cm olursa olur mu?
- Kafes uzunluğu 120 km olursa olur mu?
- 120 cm uzunluk ve 200 cm genişlik küçük kuşlar için uygun boyutlar mıdır?
- Kuşların beğeneceğini nereden bilebiliriz?
- Kuşların %90'ınıın beğenip beğenmediğini bilmek için dünyadaki tüm kuşlara bunu sormak zorunda mıyız?
- Yemek kapları plastik olabilir mi?
- Tünekler tahtadan olabilir mi?
- Kafes telleri arasındaki boşluk 2 cm olabilir mi?
- Kafes telleri arasındaki boşluk 60 cm olabilir mi?
- Kaç kuş barınabilmeli?
- Kafesin kapısının olmaması problem midir?
- Kafesin karmaşık bulmacalarla açılan 17 kapısının olması problem midir?

Yazılımların büyük bir çoğunluğu bir kuş kafesinden çok daha fazla karmaşıktır!


www.blogcmmi.com
What the customer said
What was understood
What was planned
What was developed
What was described by the business analyst
What was documented
What was deployed
The customer paid for...
How was the support
What the customer really needed

Gereksinimleri anlamanız gerekir!

- Neden? Çünkü gereksinimler beklenen davranışı açıklar
  - Beklenen davranış vs gözlenen davranış yazılım testinin temelleridir
- SRS (Software Requirements Specification) hassas bir anlaşmadır
  - Kullanıcı, geliştirici ve tester için yasal bir anlaşmadır
- Gereksinimler (genellikle) katı bir şekilde belirlenmez ve geliştirilirler
  - Ancak bütün süreç için açık bir şekilde anlaşılabilmesi oldukça önemlidir
  - QA mühendisi olarak, geliştirilmesinde katkı sunarsınız

Gereksinimler NE yapılacağını söyler, NASIL yapılacağını değil!

- **İYİ:** Sistem gelecekte gözden geçirmek üzere bütün oturumları saklamalıdır.
- **KÖTÜ:** Sistem, tüm oturumları saklamak için AllLoginsForReview adlı Singleton sınıfta ilişkisel dizi kullanacaktır.

- **İYİ:** Sistem eş zamanlı 100 kullanıcıyı desteklemelidir.
- **KÖTÜ:** Sistem eş zamanlı 100 kullanıcıyı desteklemek için BlockingQueue kullanmalıdır.

Gereksinimler NE yapılacağını söyler, NASIL yapılacağını değil! Neden?

- Kullanıcı davranışı umursar, nasıl gerçekleştiğini değil
  - Pek çok yazılım için bu genellikle doğrudur
- Geliştiricilerin uygulama implementasyonunu iyileştirmesini sağlamak için
- Kara-kutu testi implementasyon ayrıntıları verilseydi imkansız olurdu
  - Son kullanıcının gereksinimlerin geçerli olduğunu doğrulamak için kaynak koduna ihtiyacı olacaktı

Verification (doğrulama) vs Validation (sağlama)

- Verification –***Yazılımı doğru*** geliştirdik mi?
  - Yazılım ile gereksinimlerin sağlandığı emin olunur (test)
  - Beklenilmedik hataların bulunmadığından, gözlenen davranışların gereksinimleri karşıladığından ve bütün corner durumların ele alındığından emin olunur
- Validation –***Doğru yazılımı*** geliştirdik mi?
  - Yazılım müşterinin/kullanıcının gerçek istediğini sağlar mı
  - Gereksinimlerin tutarsızlık içermediğinden emin olunur
  - Gereksinimlerin doğrulanabilir olduğundan emin olunur

Gereksinim Doğrulaması

- Kodun ilk satırını yazmadan önce gerçekleşmelidir
- Gereksinimler değiştikçe, geliştirme aşamasında gerçekleşmeye devam etmelidir
- Gereksinim doğrulama bakış açıları:
  - Geçerlilik kontrolü –SRS, kullanıcı ihtiyaçları ile uyumlu mu?
  - Tamamlanma kontrolü –SRS yazılımın bütün açılarını kapsıyor mu?
  - Tutarlılık kontrolü–SRS tutarlı mı? Herhangi bir çelişki/çatışma var mı?
  - Gerçekçilik kontrolü –SRS teknoloji/bütçe/teslimat tarihi açılarından gerçekçi mi?
  - Belirsizlik kontrolü– SRS yorumlamaya açık bir yer barındırıyor mu?
  - Doğrulanabilirlik kontrolü – Gereksinimler test edilebilir mi?

Geçerlilik Kontrolü

- Gereksinimler paydaşların ihtiyaç ve gereksinimleri ile uyumlu olmalıdır
- Yaygın Yanlış Anlamalar
  - Yanlış Anlama: “Fazlası iyidir” – daha fazla fonksiyonellik, daha fazla mod vs. Gerçek: “Az fazladır” – kolay kullanım ve zarafet çok daha iyidir
  - Yanlış Anlama: Paydaşlar kullanıcılarla sınırlıdır
    Gerçek: Paydaşların içerisinde kullanıcılar, operatörler, yöneticiler, yatırımcılar vs. bulunur
  - Yanlış Anlama: Paydaşlar neye ihtiyaçları olduğunu ve ne istediklerini bilirler
    Gerçek: Kağıt üzerinde görünen şey genelde gerçek hayatta fiyaskodur.

Tamamlanma Kontrolü

- Gereksinimler sistemi bütün açılardan kapsamalıdır
  - Kapsanmamış herhangi bir şey farklı yoruma tabidir
  - Bir şeyin belli bir yoldan gerçekleştirilmesini istiyorsanız, belirtmelisiniz

Tutarlılık Kontrolü

- Gereksinimler tutarlı olmalıdır
  - Gereksinimler birbiriyle zıt olmamalıdır.
- Örnek
  - Ger. 1: "Sistem **-20 C** olursa kapanmalıdır“
  - **KÖTÜ** Ger 2: "Sistem **-40 C** veya daha soğuk olduğunda LOWTEMP uyarı ışığını yakar."
  - **İYİ** Ger 2: " Sistem **0 C** veya daha soğuk olduğunda LOWTEMP uyarı ışığını yakar."

Gerçekçilik Kontrolü

- Gereksinimler teknoloji/bütçe/teslimat tarihleri açısından gerçekçi olmalıdır
- Örnek
  - **Kötü:** Sistem Dünya ve Mars arasında 25ms gecikme ile iletişim kurabilmelidir (**Işık hızından** daha hızlı!).
  - **İYİ:** Dünya ve Mars arasındaki iletişim en uzak noktada 42 dakika en yakın noktada 24 dakikadan daha az gecikmeli olmalıdır.
- Işık hızından daha hızlı iletişim kurmak gerçekçi değil midir?
  - Quantum sinyallerinin ışık hızından daha hızlı hareket ettiği bilinmektedir
  - Ancak ardından bu soru bütçe dahilinde belirtilen tarihte teslim edilebilir mi olarak değişmektedir

Belirsizlik Kontrolü

- Gereksinimler yoruma açık olmamalıdır
- Örnek
  - **KÖTÜ**: Geçersiz bir Date tipinde veri gelirse varsayılan değer atanır.
  - **İYİ:** Geçersiz bir Date tipinde veri gelirse 1 Jan 1970 değeri atanır. (1 Jan 1970 Unix ve Linux sistemlerde 0'a karşılık gelen tarihtir.)
- Örnek
  - **KÖTÜ**: Hata durumunda sistem düzgün bir şekilde kapanacaktır.
  - **İYİ:** Sistem hata durumunda sırada bekleyen bütün komutları bir ayar dosyasına kaydedip 5 saniye içerisinde kapanacaktır.

Doğrulanabilirlik Kontrolü

- Gereksinimler test edilebilir olmalıdır.
  - Test edilebilir düzeyde belgelendirilmelidir
  - Verilen bütçe ve teslimat süresi içerisinde test edilebilir olmalıdır
- Örnek
  - **KÖTÜ**: Hesap makinesi sistemi harika olmalıdır.
  - **İYİ**: Hesap makinesi sistemi toplama, çıkarma, çarpma, bölme işlemlerini MININT ve MAXINT sınırları içerisindeki 2 sayı için yapabilmelidir.
- Örnek
  - **KÖTÜ:** Sistem, 4.137 yıl içinde 100 TB'lık bir veri setini işleyecektir.
  - **İYİ:** Sistem 1 MB veri setini 4 saat içerisinde işleyecektir.

Fonksiyonel Gereksinimler VE Kalite Özellikleri (Fonksiyonel Olmayan Gereksinimler)

- **Fonksiyonel Gereksinimler**
  - Sistemin fonksiyonel davranışlarını belirtin.
  - Sistem X'i yapmalıdır [Y şartları altında].
- **Kalite Özellikleri**
  - Sistemin bir davranışını değil genel niteliklerini belirtin.
  - Sistem X olmalıdır [Y şartları altında].
- Not: "yapmalı" vs "olmalı" ayrımdır!

Fonksiyonel Gereksinim Özellikleri

- **Ger. 1:** Sorgu sonucu bir sonuç bulunmazsa sistem "NONE" string değeri döner.

- **Ger. 2:** İç basınç 100 PSI'ı aşarsa HIPRESSURE ışığı yanar.

- **Ger. 3:** İç basınç 5 saniyeden daha fazla 100 PSI altında kalırsa HIPRESSURE ışığı söner.

Kalite Özelliği Örnekleri

- **Ger 1** – Sistem yetkisiz erişime karşı korunacaktır.
- **Ger 2** – Sistem %99.9999 ayakta olacak ve erişilebilecektir.
- **Ger 3** – Sistem kolaylıkla genişletilebilir ve bakım yapılabilir olacaktır.
- **Ger 4** – Sistem diğer işlemci mimarileriyle uyumlu olacaktır.

Bazı Kalite Özellik Kategorileri

- Reliability (Güvenilirlik)
- Usability (Kullanılabilirlik)
- Accessibility (Erişilebilirlik)
- Performance (Performans)
- Safety (Güvenlik)
- Supportability (Desteklenebilirlik)

“-ility” gereksinimleri!

Kalite özelliklerini test etmek çoğu zaman fonksiyonel özellikleri test etmekten daha zordur.

- Neden?

- Oldukça öznel olabilir
- Ölçüm zordur
- Özellikleri karşılayıp karşılamadığını belirlemek için standart kurallar yoktur

Çözüm

Kaliteyi sağlayan ölçülebilir gereksinimler konusunda paydaşlarla anlaşmaya varın.

Kaliteyi Miktar ile Belirtme

- **Performans:** saniye başına işlem sayısı, cevap zamanı
- **Güvenilirlik:** İki hata arasındaki ortalama zaman
- **Gürbüzlük**: Sistem kaç hatayı kaldırabilir?
- **Taşınabilirlik:** Desteklenecek sistem sayısı ve bunları desteklemek için gerekli zaman
- **Güvenlik:** Yıl başına düşen kaza sayısı
- **Kullanılabilirlik:** Eğitim için gerekli ortalama zaman
- **Erişilebilirlik**: Sistemi kullanabilir kişi sayısının popülasyona göre yüzdesi

Kaliteyi Miktar ile Belirtme (Örnek)

- Kalite özellikleri miktar ile belirtilebilir olmalıdır
  - Yoksa belirsizlikler ortaya çıkar
- Örnek
  - **KÖTÜ:** Sistem çok kullanışlı olmalıdır.
  - **İYİ:** Kullanıcıların %90'ından rfazlası 1 saatlik eğitimin ardından hiçbir soru sormadan kullanabilir olmalıdır.
- Örnek
  - **KÖTÜ:** Sistem bir uzay istasyonunda kullanılabilir ölçüde güvenilir olmalıdır.
  - **İYİ:** Sistemin hatalar arası ortalama zamanı 100 yıl olmalıdır.

Textbook Chapters 5'i okuyunuz

- Daha fazlası ilginizi çektiyse:

  IEEE Recommended Practice for
  Software Requirements Specifications (IEEE Std 830-1998)

- kaynaklar/IEEE830.pdf içerisinde bulunmaktadır.