Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

19. Güvenilirlik (Reliability)

Bug, Fault ve Failure

Bug (fault):

Teslim edilen sistemin spesifikasyona uymadığı programlama veya tasarım hatasıdır (örneğin, kodlama hatası, protokol hatası)

Failure:

Yazılım, kullanıcının beklediği hizmeti sunmamasıdır (örneğin, gereksinimlerde hata, kafa karıştırıcı kullanıcı arayüzü)

Bug (Hata) ve Feature (Özellik)

That's not a bug. That's a feature!

Kullanıcılar genellikle bir programın amaçlandığı gibi davranmasına rağmen yanlış olduğunu düşündükleri bir şekilde davrandığını bildirirler.

That’s not a bug. That’s a failure!

Bunun değiştirilmesi gerekip gerekmediğine karar geliştiriciler tarafından değil, müşteri tarafından verilmelidir.

Terminoloji

Hatadan kaçınma (fault avoidance)

Hatasız (bug-free) yazılım oluşturma hedefiyle oluşturulan sistemler.

Hata tespiti (test ve doğrulama) (fault detection)

Sistem devreye alınmadan önce hataların (bug) tespit edilmesi veya piyasaya sürüldükten sonra keşfedilmesi.

Hata toleransı (fault tolerance)

Sorunlar (hatalar, aşırı yüklenmeler, hatalı veriler vb.) oluştuğunda çalışmaya devam eden sistemler.

Başarısızlıklar: Bir Vaka Çalışması

İçinde 1.509 kişinin bulunduğu bir yolcu gemisi, Massachusetts, Nantucket Adası yakınlarındaki bir sığlıkta karaya oturmuştur. O sırada gemi, memurların düşündükleri yerden yaklaşık 17 mil uzaktadır. Gemi Bermuda'dan Boston'a gidiyordu.

Örnek Olay İncelemesi: Analiz

Ulusal Ulaşım Güvenliği Kurulu'nun raporundan:

- Gemi, Küresel Konumlandırma Sisteminden (GPS) gelen konum bilgilerine dayanan bir otopilot tarafından yönlendirilmekteydi.
- GPS uydulardan bir konum elde edemediği durumlarda Dead Reckoning'e (bilinen bir noktadan kat edilen mesafe ve yön) dayalı tahmini bir konum belirledi.
- GPS, Bermuda'dan ayrıldıktan bir saat sonra başarısız oldu.
- Mürettebat ekranda uyarı mesajını göremedi (veya aletleri kontrol edemedi).
- 34 saat ve 600 mil sonra, Dead Reckoning hatası 17 mil idi.

Örnek Olay İncelemesi: Yazılım Dersleri

Tüm yazılımlar belirtildiği gibi çalışmıştı (hata yoktu), ancak...

- GPS yazılımı belirlendikten sonra, **gereksinimler** değişti (bağımsız sistem artık entegre sistemin bir parçası).
- Otopilot ve GPS üreticileri, mod değişikliklerinin iletişimi konusunda farklı **tasarım** felsefeleri benimsemiştir.
- Otomatik pilot, GPS'ten gelen mesajlardaki geçerli/geçersiz durum bitlerini tanıyacak şekilde **programlanmamıştı**.
- **Kullanıcı arayüzü** tarafından sağlanan uyarılar, mürettebatı uyarmak için yeterince dikkat çekici değildi.
- Memurlar bu ekipman konusunda uygun şekilde **eğitilmemişti**.

Güvenilir yazılım, yazılım geliştirme sürecinin tüm bölümlerinin iyi bir şekilde yürütülmesini gerektirir.

Güvenilir Yazılım için Temel Faktörler

- Kalite bekleyen firma **kültürü**. Bu, yönetimden ve üst düzey teknik personel ile gerçekleşir.
- **Gereksinimler** konusunda kesin, açık anlaşma.
- **Karmaşıklığı gizleyen** tasarım ve uygulama (örneğin, yapılandırılmış tasarım, nesne yönelimli programlama).
- Basitliği, okunabilirliği ve tehlikeli yapılardan kaçınmayı vurgulayan **programlama stili.**
- Hataları kısıtlayan veya tespit eden **yazılım araçları** (örneğin, strongly-typed diller, kaynak kontrol sistemleri, hata ayıklayıcılar).
- Gereksinimler, sistem mimarisi, program tasarımı, uygulama ve kullanıcı testi dahil olmak üzere geliştirmenin tüm aşamalarında sistematik **doğrulama**.
- **Değişikliklere** ve bakıma özellikle dikkat edilmesi.

Güvenilir Sistemler Oluşturma: Firma Kültürü

İyi firmalar iyi sistemler yaratır:

- Yöneticiler ve üst düzey teknik personel örnek teşkil etmelidir.
- Grubun çalışma tarzını herkesin kabul etmesi gerekir (örneğin, toplantılar, hazırlık, junior geliştiriciler için destek).
- Görünürlük.
- Bir sonrakine geçmeden önce bir görevin tamamlanması (örneğin, belgeler, koddaki yorumlar).

Güvenilir Sistemler Oluşturma: Firma Kültürü

Örnek: bir kütüphane sistemi

Problem:

- Veritabanı sürekli çökmekteydi ve veri kaybı oluşuyordu.
- Art arda gelen sürümler sorunu çözemedi.

Analiz:

- Takımın iyi bir teknik planı vardı ancak zamana da ihtiyacı vardı.
- Üst yönetim, sistemin hazır olmadan önce yayınlanmasında ısrar etti.

Düzeltme (Fix):

- Ekibe zaman tanıyın.
- Üst yönetimi değiştirin.

Güvenilir Yazılım Oluşturma: Kalite Yönetim Süreçleri

Varsayım:

İyi yazılımlar, iyi süreçler olmadan mümkün değildir

Rutinin önemi:

Standart terminoloji (gereksinimler, tasarım, kabul vb.)

Yazılım standartları (kodlama standartları, isimlendirme kuralları vb.)

Komple sistemin düzenli olarak oluşturulması (build) (genellikle günlük)

Dahili ve harici belgeler

Raporlama prosedürleri

Bu rutin hem heavyweight hem de lightweight geliştirme süreçleri için önemlidir.

Güvenilir Yazılım Oluşturma: Kalite Yönetim Süreçleri

Zaman kısa olduğunda...

Sürecin ilk aşamalarına daha fazla dikkat edin: fizibilite, gereksinimler, tasarım.

Gereksinimler sürecinde hatalar yapılırsa, bunları daha sonra düzeltmek için çok az zaman olacaktır.

Deneyler, erken aşamalarda fazladan zaman ayırmanın genellikle yayınlama süresini azaltacağını göstermektedir.

Güvenilir Yazılım Oluşturma: Müşteri ile İletişim

Müşterinin ihtiyaçlarını karşılamıyorsa bir sistem işe yaramaz

- Müşteri, üzerinde anlaşmaya varılan gereksinimleri ayrıntılı olarak anlamalı ve gözden geçirmelidir.
- Müşteriye bir şartname belgesi sunmak ve imzasını istemek yeterli değildir.
- Müşteri personelinin uygun üyeleri, tasarımın ilgili alanlarını (operasyonlar, eğitim materyalleri, sistem yönetimi dahil) gözden geçirmelidir.
- Kabul testleri müşteriye ait olmalıdır.

Güvenilir Yazılım Oluşturma: Karmaşıklık

İnsan zihni yalnızca sınırlı karmaşıklığı kapsayabilir:

- Anlaşılabilirlik
- Basitlik
- Karmaşıklığın bölümlenmesi

Basit bir bileşeni doğru yapmak, karmaşık bir bileşenden daha kolaydır.

Güvenilir Yazılım Oluşturma: Değişim

Değişiklikler kolayca sorunlara yol açabilir

Değişiklik yönetimi

- Kaynak kodu yönetimi ve sürüm kontrolü
- Değişiklik isteklerinin ve hata raporlarının takibi
- Gereksinimleri, spesifikasyonları, tasarımları ve diğer belgeleri değiştirme prosedürleri
- Regresyon testi (daha sonra tartışılacaktır)
- Yayımlama (release) kontrolü

Yeni işlevler eklerken veya hataları düzeltirken, sistem mimarisini veya genel program tasarımını ihlal eden yamalar yazmak kolaydır. Bundan mümkün olduğunca kaçınılmalıdır. Yüksek kaliteli bir sistemi korumak için mimariyi değiştirmeye hazır olun.

Güvenilir Yazılım Oluşturma: Hata Toleransı

Amaç:

Sorunlar oluştuğunda çalışmaya devam eden bir sistem.

Örnekler:

- Geçersiz giriş verileri (örneğin, bir veri işleme uygulamasında)
- Aşırı yük (örneğin, ağa bağlı bir sistemde)
- Donanım arızası (örneğin, bir kontrol sisteminde)

Genel Yaklaşım:

- Arıza tespiti
- Hasar değerlendirmesi
- Arıza kurtarma
- Arıza onarımı

Hata Toleransı: Kurtarma

Geriye dönük kurtarma

- Belirli olaylarda (kontrol noktaları) sistem durumunu kaydedin. Hatadan sonra, son denetim noktasında durumu yeniden oluşturun.
- Kontrol noktalarını, son kontrol noktasından işlemlerin otomatik olarak tekrarlanmasını sağlayan **sistem günlüğü (işlemlerin denetim izi)** ile birleştirin.

Kurtarma yazılımının test edilmesi zordur.

Güvenilir Yazılım Oluşturma: Küçük Ekipler ve Küçük Projeler

Küçük ekipler ve **küçük projeler** güvenilirlik açısından avantajlara sahiptir:

- Küçük grup iletişimi, ara dokümantasyon ihtiyacını ve yanlış anlamaları azaltır.
- Küçük projelerin test edilmesi ve güvenilir hale getirilmesi daha kolaydır.
- Küçük projelerin geliştirme döngüleri daha kısadır. Gereksinimlerdeki hataların düzeltilmesi daha az olasıdır ve daha ucuzdur.
- Bir proje tamamlandığında, bir sonrakini planlamak daha kolaydır.

Geliştirilmiş güvenilirlik, çevik geliştirmenin son birkaç yılda popüler hale gelmesinin nedenlerinden biridir.

Güvenilirlik Metrikleri

Güvenilirlik

Operasyonel kullanımda meydana gelen bir arıza olasılığı.

Çevrimiçi sistemler için geleneksel ölçümler

- Arızalar arasındaki ortalama süre (Mean time between failures)
- Kullanılabilirlik (çalışma süresi)
- Ortalama onarım süresi

Piyasa Ölçümleri

- Şikayet
- Müşteriyi elde tutma

Dağıtık Sistemler için Güvenilirlik Metrikleri

Geleneksel metriklerin çok bileşenli sistemlerde uygulanması zordur:

- Mükemmel ortalama güvenilirliğe sahip bir sistem, belirli kullanıcılara korkunç hizmet verebilir.
- Büyük bir ağda, herhangi bir anda bir şey sorun çıkaracak, ancak çok az kullanıcı bunu görecektir.
- Çok sayıda bileşen olduğunda, sistem yöneticileri sorunlu alanları belirlemek için otomatik raporlama sistemlerine güvenmektedir.

Metrikler: Kullanıcıların Güvenilirlik Algısı

Algılanan güvenilirlik şunlara bağlıdır:

- Kullanıcı davranışı
- Girdi seti
- Hatanın yol açtıkları

Kullanıcı algısı, hataların dağılımından etkilenir

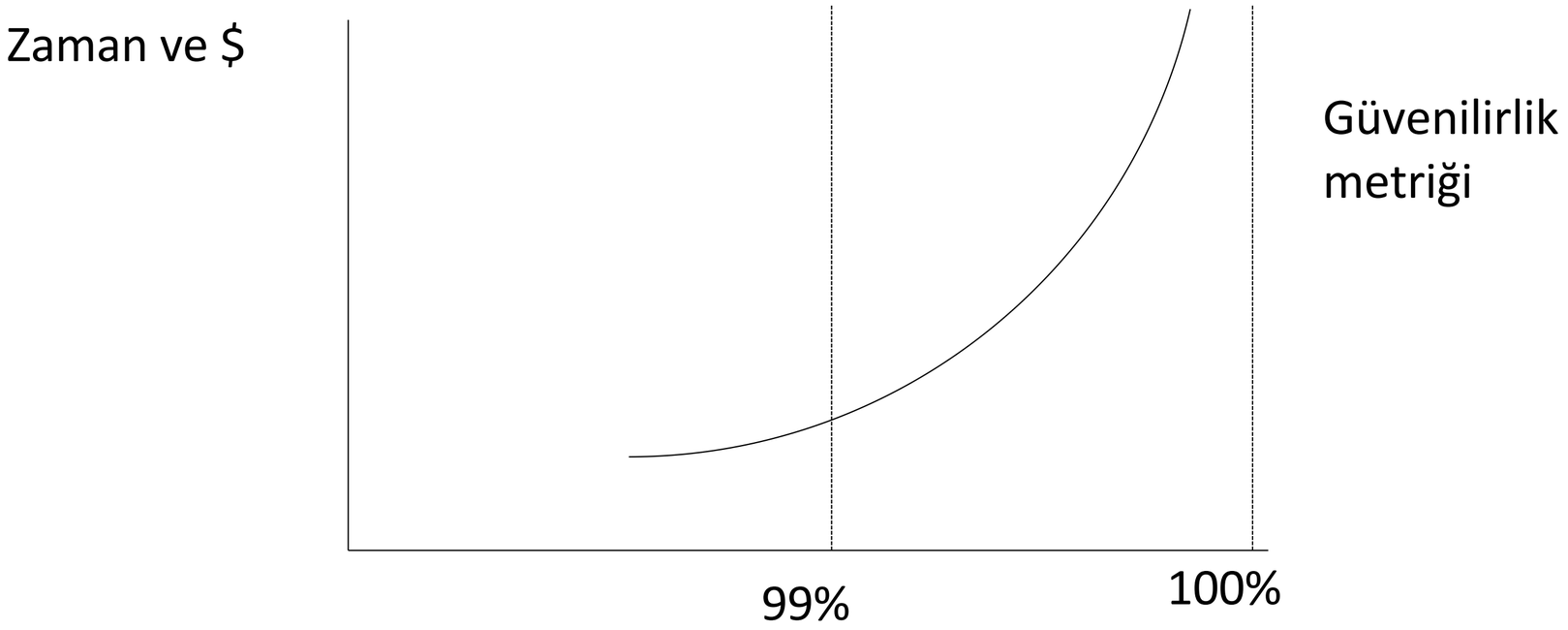
- Sık sık çöken bir kişisel bilgisayar veya birkaç yılda bir iki gün hizmet dışı kalan bir makine.
- Sık sık çöken ancak veri kaybı olmadan hızlı bir şekilde geri gelen bir veritabanı sistemi veya üç yılda bir başarısız olan ancak verilerin yedekten geri yüklenmesi gereken bir sistem.
- Başarısız olmayan ancak çok yavaş çalıştığında öngörülemeyen dönemleri olan bir sistem.

Gereksinimler için Güvenilirlik Ölçümleri

Örnek: ATM kart okuyucu

Hata sınıfı	Örnek	Metrik (gereksinim)
Kalıcı bozulmayan	Sistem herhangi bir kartla çalışamıyor -- yeniden başlat	1.000 günde 1
Geçici Bozulmayan	Sistem hasarsız bir kartı okuyamıyor	1,000 işlemde 1
Bozulma	Bir işlem modeli finansal veritabanını bozuyor	Hiç

Metrikler: Geliştirilmiş Güvenilirliğin Maliyeti



Örnek.

Birçok süper bilgisayar günde ortalama 10 saat üretken çalışma yapar. Güvenilirliği artırmak için paranızı nasıl harcıyorsunuz?

Örnek: Merkezi Hesaplama Sistemi

Merkezi bir bilgisayar sistemi (örneğin, bir sunucu çiftliği) tüm kuruluş için hayati önem taşır (örneğin, bir İnternet alışveriş sitesi). Herhangi bir başarısızlık ciddidir.

1. Adım: Her hatayla ilgili verileri toplayın

- Her hatayı kaydeden bir veritabanı oluşturun
- Her hatayı analiz et:

  Donanım yazılımı (default)

  çevre (örn. güç, klima)

  insan (örneğin, operatör hatası)

Örnek: Merkezi Hesaplama Sistemi

2. Adım: Verileri analiz edin

- Haftalık, aylık ve yıllık istatistikler
  - Arıza ve kesinti sayısı
  - Ortalama onarım süresi
- Bileşenlere göre trendlerin grafikleri, örn.,
  - Disk sürücülerinin arıza oranları
  - Elektrik kesintilerinden sonra donanım arızaları
  - Her bileşendeki yazılım hatalarından kaynaklanan çökmeler
  - İnsan hatası kategorileri

Örnek: Merkezi Hesaplama Sistemi

Adım 3: Kaynakları, faydanın maksimum olacağı yerlere yatırın, örneğin,

- Yazılım iyileştirmeleri için öncelik sırası
- Operatörler için değiştirilen prosedürler
- Yedek donanım
- Elektrik kesintisinden sonra düzenli olarak yeniden başlatma

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

19. Güvenilirlik (Reliability)

Ders Sonu