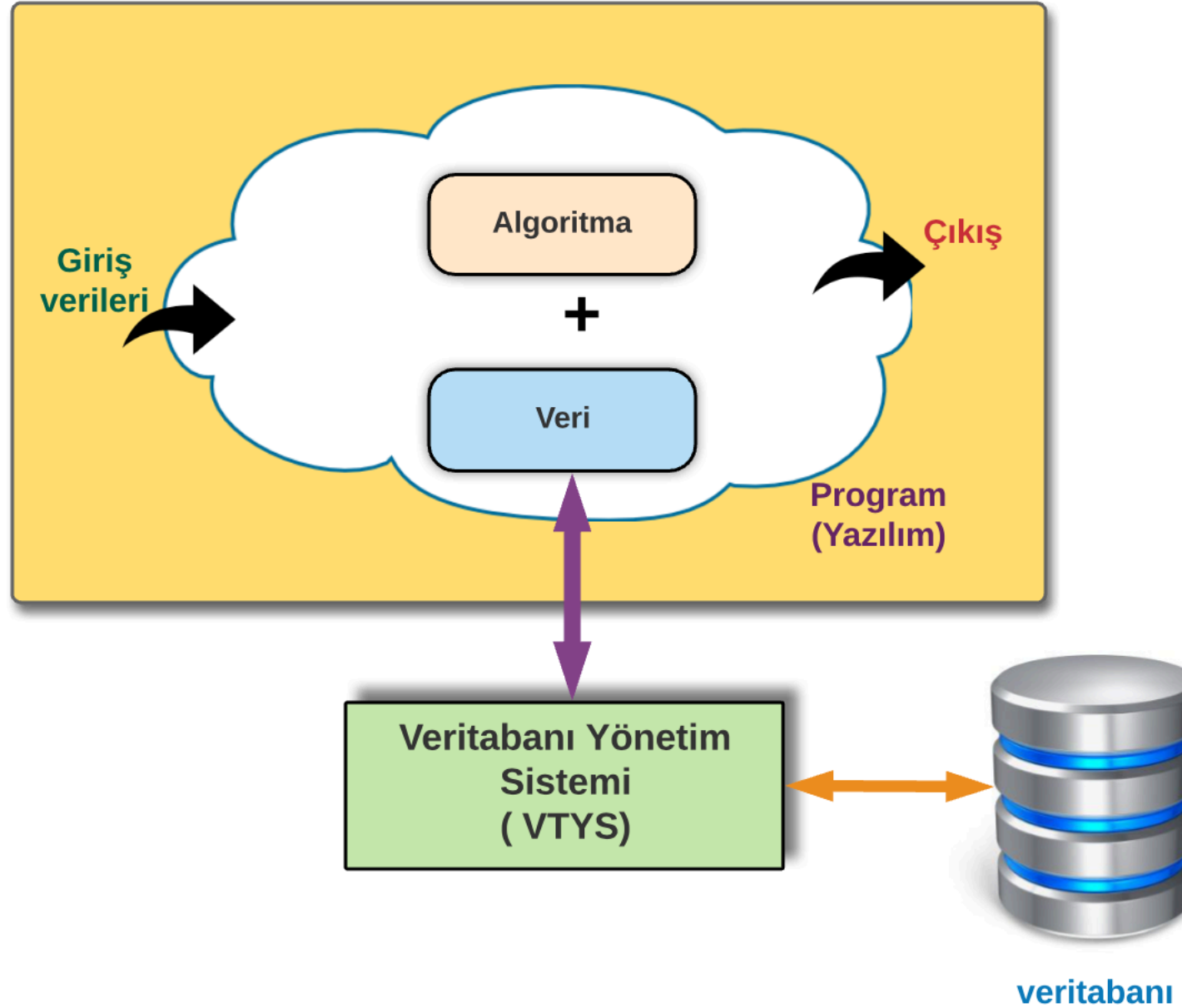


Nesne Yönelimli Analiz ve Tasarım

Yazılım Geliştirme Yaşam Döngüsü

Büyük Resim

Gerçek Dünya
Problemi



İyi Yazılım?

iyi bir yazılım;

- * son kullanıcı açısından bakıldığında,
 - * beklendiği gibi çalışan, istenen fonksiyonları yerine getiren, hatalı sonuçlar vermeyen yazılım
- * yazılım geliştiriciler açısından,
 - * sonradan değişikliği kolaylaştıran
 - * kod tekrar kullanımının fazla olduğu
 - * yeni özellik kazandırmanın kolay olduğu
 - * temiz kodlanmış, anlaşılabilir, karmaşıklığı azaltılmış yazılımlardır
 - * bunu sağlamanın yolu ise tasarım ilkelerinin ve desenlerinin uygun kullanımına bağlıdır.

Yazılım Geliştirme Yaşam Döngüsü

YGVD içerisinde 5 temel evre bulunmaktadır.

Günümüzde farklı yazılım geliştirme yaşam döngüsü modelleri (software processes) bulunmaktadır: waterfall, spiral, iterative/ incremental, agile, unified vb...

bu modellerin tamamı, bir şekilde bu evreleri içerir.

Nasıl yapmalı?

Tasarım ilkeleri ve Tasarım Desenleri kullanarak modüllerin bir araya getirilmesi
UML ile Modelleme yapılması

Test et

Fonksiyon, başarımlar ve güvenlik testleri

1

Analiz (Analysis)

2

Tasarım (Design)

3

Gerçekleme
(Implementation)

4

Test (Test)

5

Bakım (Maintenance)

Gereksinimlerin Belirlenmesi
Kısıtların Belirlenmesi
Nesneler ve aralarındaki bağıntılar belirlenmesi
UML ile Modelleme yapılması

Geliştirilen modellerin uygun bir programlama dili ile kodlanması

Yazılımın hedef sisteme konuşlandırılması (deployment)
Sonradan değişiklik

Yap

Kullan

Ne yapmalı?

1. Analiz

- Geliştirilecek yazılımın gereksinimleri belirlenir.
- Bu adım proje yöneticisi, iş analisti ve uygulama programı yöneticisi nin sorumluluğundadır. Uç kullanıcılar, yöneticiler ve geliştirilecek yazılımla ilgili diğer paydaşlarla görüşülerek gereksinimler belirlenir (requirements elicitation). Aşağıdaki sorulara yanıtlar aranır
 - Sistem ne yapmalı? ne yapması bekleniyor?
 - sistemi kimler, ne yapmak için kullanacak?
 - giriş verileri neler?
 - çıkış verileri neler?
 - ne tür veriler ele alınacak/işlenecek (telefonNo, TC kimlikNo ... Bu verilere bakılarak veritabanı tasarımı için ihtiyaç duyulan iş kuralları da oluşturulur)
 - takip edilmesi gereken standartlar, kurallar, yönetmelikler, güvenlik gereksinimleri v.s. neler?

- Geliştirilecek sistemle ilgili kısıtların belirlenmesi
- Uygulama Program yöneticisi nin sorumluluğundadır
 - Uygulama hangi işletim sisteminde çalışacak? işletim sistemi kısıtları neler?
 - bellek kısıtları, işlem gücü kısıtları, ağ kısıtları vb. operasyonel ya da mimariyle ilgili kısıtlar neler?

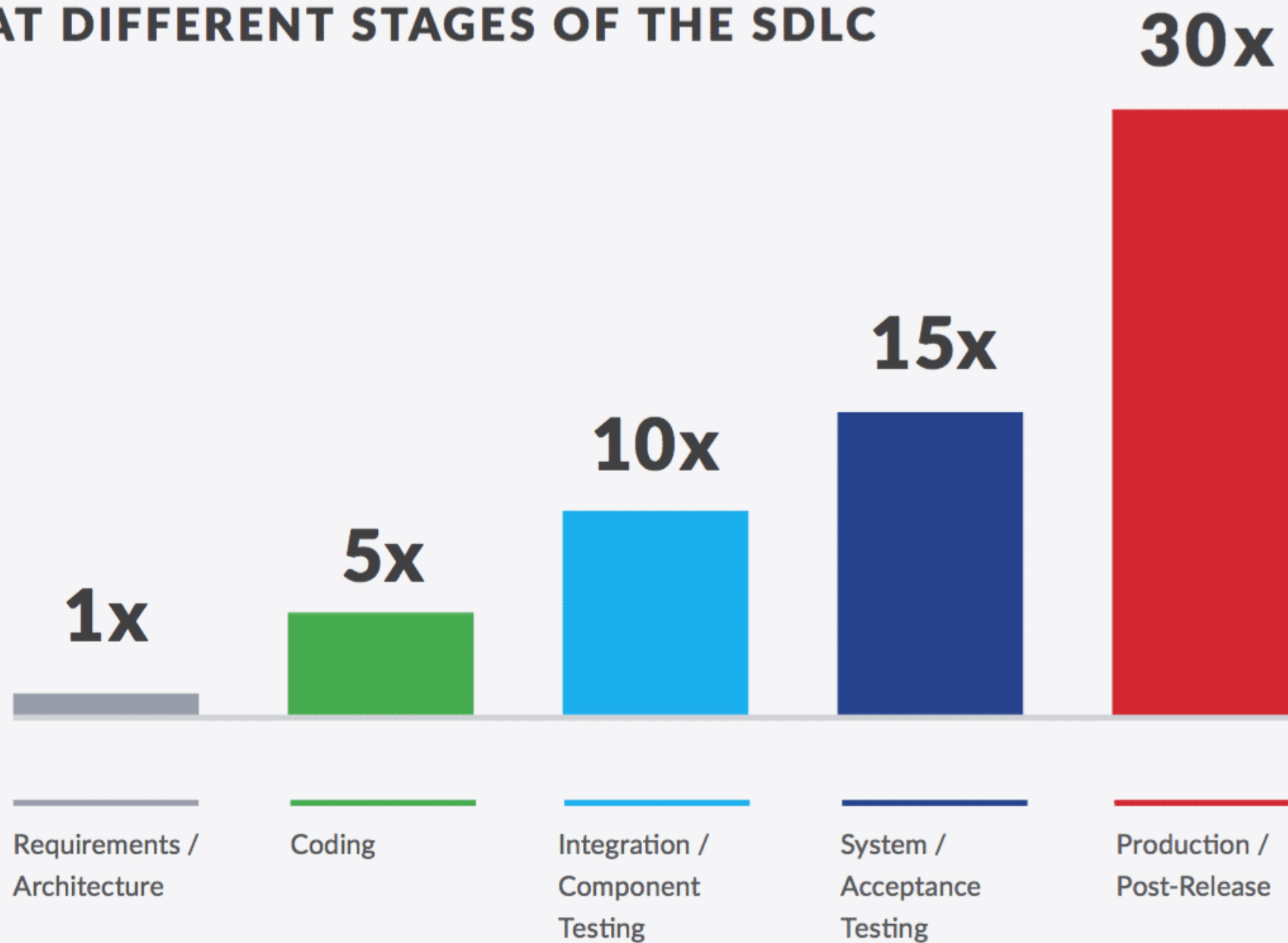
Kaynak kullanımı, hız, güvenlik gibi ihtiyaçlara uygun dillerin/teknolojilerin/algoritmaların/kütüphanelerin seçilmesi....

2. Tasarım

- Kısıtlar ihlal edilmeden yazılım gereksinimlerinin **nasıl** karşılanacağı belirlenir.
- Tasarım aşamasında oluşturulacak modeller için yaygın olarak UML (Unified Modeling Language) kullanılır.
- Yazılıma bakış açıları farklı aktörler arasındaki haberleşmeyi sağlar
- Kodlamayı kolaylaştırır. Tasarımdaki bileşenler yazılım modüllerine dönüştürülür. Bu modüllerin nasıl etkileşeceği de bu adımda belirlenir.
- Tasarım ekibi tarafından **gereksinimler** ve **mimari kısıtlar** göz önüne alınarak aşağıdakiler belirlenmelidir:
 - Kullanılacak diller ve programlama yöntemleri
 - Takip edilecek kodlama standartları (Kodlama stili, isimlendirme kuralları, kullanılmaması gereken modüller vb.)
 - Kullanılacak modüller, kütüphaneler
 - Veri yapıları, veritabanı yönetim sistemi ...
- Oluşturulan modeller gerçektenmeden önce dikkatli olarak incelenmelidir. Böylece, olası hataların düzeltilme maliyeti düşürülmüş olur.

YG'D evreleri ilerledikçe, değişiklik yapma maliyeti artar.

THE RELATIVE COST OF FIXING A FLAW AT DIFFERENT STAGES OF THE SDLC



SOURCE: NIST

<https://www.opswat.com/blog/secure-sdlc-at-opswat>

3. Gerçekleme

- Önceki aşamalarda geliştirilen modeller kodlanarak yazılım uygulamasına dönüştürülür.
- Yazılım geliştirme ekibi sorumluluğundanki bu adımda:
 - Kodlama standartlarına uyulmalı
 - Tasarım dökümanı takip edilmeli
 - kod incelemesi (code review) yapılarak kod kalitesi ve güvenlik artırılmalı.
 - Güvenlik gereksinimleri dikkate alınmalı (giriş filtreleme, çıkış kodlama vb.)

4. Test

- Geliştirilen uygulama; işlevsellik, başarımlar ve güvenlikle ilgili gereksinimler dikkate alınarak test edilir.
- Böylece uygulamanın beklendiği gibi çalışması sağlanır.
- Bu aşamada birim testi (unit testing), tümleştirme testi (integration testing), sistem testi (system testing), kabul testi (acceptance testing) yapılır.
- Güvenlik açısından sızma testleri yapılır.
- Bu aşamadan sorumlu Test ekibi aşağıdakileri kontrol etmelidir:
 - sistem kararsız çalışmamalı
 - testler kodların tamamını kapsamalı (code coverage %100)
 - sistem tüm gereksinimleri karşılamalı

5. Bakım & Konuşlandırma (Maintenance & Deployment)

- Sürüm Yöneticisi (Release Managers)
yazılımı kullanıma hazırlar
- uygulamanın kurulum işlemi
- gerekli bileşenlerin kurulumu
- Güncellemeler

5. Bakım & Konuşlandırma (Maintenance & Deployment)

- Uygulama kullanılmaya başladıktan sonra meydana gelen değişiklik gereksinimleri bu aşamada (maintenance) ele alınır.
- Değişiklik nedenleri (Lientz and Swanson):
 - Adaptive - yazılımın çalıştığı ortamın değişmesinden kaynaklanan (DBMS, OS)
 - Perfective - yeni/değişen kullanıcı isteklerinden (sistem fonksiyonlarının iyileştirilmesi) kaynaklanan
 - Corrective - kullanıcılar tarafından tespit edilen hatalardan kaynaklanan
 - Preventive - gelecekte meydana gelebilecek sorunların önlenmesini sağlamak için
- 75% adaptive ve perfective, ~%21 hataların düzeltilmesi (corrective).