Yazılım Mühendisliği Nedir?

Yazılım mühendisliği, yazılım tasarımı ve geliştirilmesine sistematik bir yaklaşım sunan bir disiplindir.
Bir yazılım mühendisi; yazılım sistemlerinin tasarımı, geliştirilmesi, bakımı, test edilmesi ve dokümantasyonundan sorumludur.
Ayrıca, paydaşlarla, üçüncü taraf tedarikçilerle, güvenlik uzmanlarıyla ve diğer ekip üyeleriyle iş birliği yaparak projenin tüm aşamalarında aktif rol oynar.

Yazılım mühendisleri sistemi bütün olarak ele alırken, yazılım geliştiriciler genellikle belirli işlevlerin uygulanmasından sorumludur.
Bu süreçte CASE (Computer-Aided Software Engineering) araçları, verimliliği artırmak ve hataları azaltmak için sıklıkla kullanılır.

Yazılım Geliştirme Yaşam Döngüsü (SDLC)

SDLC (Software Development Life Cycle), yazılım geliştirme sürecini sistematik bir şekilde tanımlayan bir modeldir.
Bu model, riskleri azaltmayı ve verimliliği artırmayı amaçlar.

SDLC genellikle 6 aşamadan oluşur:

1. **Planlama: Gereksinimlerin toplanması ve proje planının oluşturulması.**
2. **Tasarım: Sistem mimarisi ve tasarım dokümanlarının hazırlanması.**
3. **Geliştirme: Kodlama sürecinin gerçekleştiği aşama.**
4. **Test: Yazılımın hatalarının tespit edilip düzeltilmesi.**
5. **Dağıtım: Kodun üretim ortamına aktarılması.**
6. **Bakım: Geri bildirimlerin alınması, geliştirmelerin yapılması ve yeni sürümler için iyileştirme önerilerinin hazırlanması.**

Kaliteli Yazılım

Kaliteli bir yazılım, kullanıcı ihtiyaçlarını eksiksiz karşılayan ve hatasız çalışan yazılımdır.
Kalite süreci genellikle Alfa, Beta ve Genel Kullanım (General Availability) aşamalarını içerir.

- **Gereksinim Toplama: Yazılımın ihtiyaç duyulan özelliklerinin belirlenmesi ve belgelenmesi.**
- **Tasarım: Gereksinimlerin geliştiriciler için uygulanabilir bir yapıya dönüştürülmesi.**
- **Kodlama: Belirli standartlara ve iyi uygulamalara uygun şekilde yazılımın geliştirilmesi.**

- **Test: Yazılımın gereksinimlere uygunluğunun doğrulanması ve hataların giderilmesi.**

Gereksinimler (Requirements)

Gereksinim toplama süreci, yazılımın ne yapması gerektiğini anlamak için yürütülen sistematik bir çalışmadır.
Altı temel adımdan oluşur:

1. **Paydaşları Tanımlama**
2. **Hedef ve Amaçları Belirleme**
3. **Gereksinimleri Toplama**
4. **Gereksinimleri Belgeleme**
5. **Gereksinimleri Analiz Etme ve Onaylama**
6. **Gereksinimleri Önceliklendirme**

Gereksinim Belgeleri

- **SRS (Software Requirements Specification): Yazılımın işlevselliğini ve performans standartlarını tanımlar.**
- **URS (User Requirements Specification): Son kullanıcıların ihtiyaç ve beklentilerini belirler.**
- **SysRS (System Requirements Specification): Sistem yetenekleri, arayüzler ve kullanıcı özellikleri gibi geniş kapsamlı bilgileri içerir.**

Yazılım Geliştirme Metodolojileri

Yazılım geliştirme sürecinde bilgi paylaşımını ve iletişimi düzenlemek için çeşitli metodolojiler kullanılır.
En yaygın üç yaklaşım: Waterfall (Şelale), V-Shape (V-Şekli) ve Agile (Çevik)'tir.

Waterfall (Şelale) Modeli

- **Aşamalar sıralı olarak ilerler.**
- **Her aşama tamamlanmadan bir sonraki aşamaya geçilmez.**
- **Müşteri ürünü yalnızca test aşamasında görür.**

V-Shape Modeli

- **Waterfall modeline benzer, ancak her aşama için doğrulama ve geçerlilik testleri yapılır.**
- **Aşamalar: Gereksinim Analizi → Sistem Tasarımı → Mimari Tasarım → Modül Tasarımı → Kodlama → Testler (Birim, Entegrasyon, Sistem, Kabul).**

Agile (Çevik) Metodolojisi

- **Kısa döngüler (Sprint) halinde tekrarlanan bir geliştirme sürecidir.**
- **Her sprint sonunda çalışan bir ürün parçası teslim edilir ve geri bildirim alınır.**
- **Sürekli iyileştirme ve değişen gereksinimlere hızlı uyum sağlanır.**

Yazılım Sürümleri

Yazılım sürüm numaraları, yapılan güncellemeleri ve değişiklikleri belirtir.
Genellikle Major.Minor.Patch.Build formatında yazılır.

- **Birinci sayı (Major): Büyük değişiklikleri gösterir.**
- **İkinci sayı (Minor): Küçük eklemeleri ifade eder.**
- **Üçüncü sayı (Patch): Hata düzeltmelerini belirtir.**
- **Dördüncü sayı (Build): Derleme veya tarih bilgisini gösterebilir.**

Uyumluluk sorunlarını önlemek için sürüm yönetimi düzenli olarak yapılmalıdır.

Yazılım Testi

Yazılım testi, yazılımın gereksinimlere uygunluğunu kontrol ederek kaliteyi güvence altına alır.
Amaç, hataları erkenden tespit edip yazılımın güvenilirliğini sağlamaktır.

Test Türleri

- **Fonksiyonel Test: Yazılımın belirli işlevlerini test eder. (Örnek: "Sepete ekle" işlemi)**
- **Fonksiyonel Olmayan Test: Performans, güvenlik ve ölçeklenebilirlik gibi nitelikleri değerlendirir.**
- **Regresyon Testi: Yeni değişikliklerin mevcut işlevselliği olumsuz etkileyip etkilemediğini kontrol eder.**

Test Seviyeleri

1. **Birim Testi: Kodun küçük parçaları test edilir.**
2. **Entegrasyon Testi: Modüllerin birlikte çalışması değerlendirilir.**
3. **Sistem Testi: Tüm sistemin gereksinimlere uygunluğu test edilir.**
4. **Kabul Testi: Yazılımın kullanıcı ihtiyaçlarını karşılayıp karşılamadığı kontrol edilir.**

Yazılım Dokümantasyonu

Yazılım dokümantasyonu, yazılımın ne yaptığını ve nasıl kullanılacağını anlatan bilgi bütünüdür.
Yazılı, görsel veya video formatında hazırlanabilir ve sürekli güncel tutulmalıdır.

Dokümantasyon Türleri

- **Ürün Dokümantasyonu: Yazılımın işlevselliğiyle ilgilidir.**
- **Süreç Dokümantasyonu: Bir görevin nasıl tamamlanacağını açıklar.**
- **SOP (Standart İşlem Prosedürü): Belirli görevlerin nasıl yapılacağını detaylı biçimde tanımlar.**

Ürün dokümantasyonu; gereksinimler, tasarım, teknik detaylar, kalite güvencesi ve kullanıcı kılavuzu gibi bölümleri içerir.

Yazılım Projesindeki Roller

Bir yazılım projesinde birçok farklı rol bulunur ve her biri sürecin farklı bir yönünden sorumludur:

- **Proje Yöneticisi / Scrum Master: Proje ilerleyişini ve ekip içi iletişimi yönetir.**
- **Paydaşlar: Ürünün hedef kitlesini temsil eder ve gereksinimleri belirler.**
- **Sistem Mimarları: Yazılımın genel mimarisini tasarlar.**
- **UX Tasarımcıları: Kullanıcı deneyimini ve arayüz tasarımını oluşturur.**
- **Geliştiriciler: Yazılımı kodlar ve uygulamayı geliştirir.**
- **Test Uzmanları (QA Mühendisleri): Yazılımın kalitesini kontrol eder.**
- **Site Güvenilirlik Mühendisi (SRE): Sistemlerin sürekliliğini ve güvenilirliğini sağlar.**
- **Ürün Yöneticisi: Ürünün vizyonunu ve stratejisini belirler.**
- **Teknik Yazar: Kullanıcı dokümantasyonlarını hazırlar.**

Özet

Yazılım mühendisliği; gereksinimlerin belirlenmesi, tasarım, geliştirme, test, dağıtım ve bakım aşamalarını içeren çok yönlü bir süreçtir.
Bu süreç, doğru metodoloji ve rollerin etkin iş birliğiyle yürütüldüğünde, kaliteli, güvenilir ve sürdürülebilir yazılım ürünleri ortaya çıkar.