YAZILIM KAVRAMI

Yazılım en yalın biçimiyle, *'Bir bilgisayar sisteminin donanım bileşenleri dışında kalan her şey'* olarak tanımlanabilir. Elektrik-Elektronik Mühendisleri Enstitüsü (IEEE) ise yazılımı; 'Bir bilgisayar sisteminin işleyişi ile ilgili kodların, prosedürlerin, verilerin ve dokümanların oluşturduğu bütün' olarak tanımlar.

YAZILIM KRİZİ

1965-1980 yılları üniversite ve iş alanlarında güçlü Bilgisayar Merkezleri (Computer Centers)'in kurulduğu, bilimsel ve iş alanı uygulamalarının ayrıldığı ve özellikle kapsamlı iş alanı uygulamalarının bilgisayar sistemlerine uyarlanmaya başladığı, Yazılım Evlerinin (Software Houses) yavaşta olsa geliştiği yıllar olarak tanımlanabilir. Başta IBM olmak üzere, yazılımın kiralanabilmesi ya da satın alınması döneminin başladığı bu yılların bir özelliği; uygulamaların ve özel koşullarına yönelik uygulama problemlerine ve özellikle "Büyük Uygulama Projelerine" bütüncü ortak çözüm arama girişimi görülmeye başlaması için önemlidir.

1970'li yılların ikinci yarısından başlayan, yazılım geliştirme sayısında ve boyutunda sıçrama döneminden söz edilebilirse de; büyük-proje konusunda gerek yazılım evlerinin, gerekse kurulu Bilgisayar Merkezleri uzmanlarının çözümleme, tasarım, programlama gibi temel aşamalarda karşılaştığı sorunlar nedeni ile uygulamayı öngörülen sürede ve maliyetle tamamlayamamaktan doğan, maddi ve manevi zararlar başta iş sahiplerini olduğu kadar, çalışanları da güç durumda bırakmıştır. Bu süreç, kaçınılmaz olarak; **'Yazılım Bunalımı (Software Crisis)'** adı verilen bir bunalım oluşumu yaratmıştır.

Olumsuzluğun en önemli yanı; bilgisayar uygulamasına binlerce hatta milyonlarca ABD doları yatıran yöneticilerin yatırımlarının karşılığındaki hayal kırıklığı olmuştur. Sorun, donanımdan çok yazılım tasarımının başarısızlığından ya da denenen çözümlerin ya da deneyimin yetersiz olmasından kaynaklandığı çok açıktı.

Tarihsel sürece baktığımızda, kişisel çabalar yanında, bunalıma çözüm arama konusunda bütüncül çabanın Avrupa'dan geldiğini görülmüştür. İlk ve ses getirici uluslararası girişim ve çalışma; 1968 Yılında Almanya'nın Garmish-Partenkirchen kentinde, NATO Bilim Kurulunun desteklediği Münih Teknik Üniversitesi (München Technischen Universitat)'nin ev-sahipliğinde; İnformatik Enstitüsü Başkanı Bilgisayar Bilimcisi Prof. Dr. Friedrich L. Bauer yönetimindeki konferanstır. Konferansa, 50 matematikçi, mühendis ve bilgisayar bilimcisi

katılmış ve "Büyük Yazılım Projesi" geliştirme ve yöntemleri konusu ele alıp, çözümler tartışılmıştır.

Bu bağlamda, araştırma ve uygulamacıların çözüm önerileri ışığında, büyük yazılım projelerinin yazılım çözümleme, tasarım ve yönetimi için, kuramsal farklı yaklaşımlara gerek olduğunu ortaya konmuş, tartışılmış ve sonucunda, özetle konunun Yazılım Mühendisliği (Software Engineering) adı altında özel ve yani bir disiplin olarak geliştirilmesi kararı alınmıştır. Böylece, yazılım bunalımına çözüm getirecek Yazılım Mühendisliğinin doğumu bu konferans olarak tarihe geçmiştir.

Literatürde yazılım mühendisliği kavramı, müşterinin ihtiyaçlarını karşılayan, öngörülmüş bütçe ile zamanında teslim edilen ve hatasız bir yazılım geliştirmek için teknik yöntemler öngören bir mühendislik disiplini olarak tanımlanmaktadır.

Yazılım mühendisliği devletin, toplumun ulusal ve uluslararası iş dünyası ve kurumların işleyişi için gereklidir. Modern dünyayı yazılım olmadan yürütemeyiz. Ulusal altyapılar ve servisler bilgisayar tabanlı sistemler tarafından kontrol edilir ve elektrikli ürünlerin çoğu bilgisayar ve kontrol yazılımı içerirler.

Yazılım mühendisliğinin tarihçesi

Yazılım mühendisliği kavramı ilk olarak 1968'de o zaman yazılım krizi (Naur ve Randell 1969) olarak adlandırılan konuları tartışmak için düzenlenen bir konferansta önerilmiştir. Program geliştirmedeki birbirinden farklı yaklaşımların büyük ve karmaşık yazılım sistemlerinin geliştirilmesinde ölçeklenmediği anlaşılmıştı. Yazılım sistemleri güvenilir olmamakta, beklenenden daha fazla maliyetle geç teslim edilmekteydi.

1970'ler ve 1980'ler boyunca, yapısal programlama, bilgi saklama ve nesneye yönelik geliştirme gibi çeşitli yeni yazılım mühendisliği teknikleri ve yöntemleri geliştirilmiştir. Günümüzdeki yazılım mühendisliğinin temeli olan araçlar ve standart gösterimler geliştirilmiştir.

Yazılım geliştirme sürecinin adımlarına uygun olarak, yazılım hata nedenleri aşağıdaki gibi sınıflandırılabilir;

- Gereksinimlerin hatalı tanımlanması,
- Müşteri-geliştirici arasındaki iletişim başarısızlığı,
- Yazılım gereksinimlerinden kasıtlı sapmalar,
- Mantıksal tasarım hataları,

- Kodlama hataları,
- Kodlama standartlarına uymama,
- Test sürecinin eksiklikleri,
- Kullanıcı arayüzü ve prosedür hataları,
- Dokümantasyon hataları.

‘İYİ YAZILIM’ ne demektir?

İyi bir yazılım ürünü, müşterinin beklediği işlevsellik ve performansın yanı sıra aşağıdaki özellikleri de taşımalıdır;

- Bakım-yapılabilirlik: Yazılım ürününün değişen müşteri gereksinimlerine göre kolaylıkla değişebiliyor olmasıdır.
- Güvenilirlik: Yazılım ürününün çökmeden işletimde kaldığı süreçte dış zorlamalara karşı güvenilir olmasıdır.
- Verimlilik: Yazılım ürünü sistem kaynaklarının israfına sebep olmamalıdır.
- Kabul edilebilirlik: Yazılım ürününün kullanıcılar tarafından anlaşılır ve kullanılabilir olarak diğer sistemlerle uyumlu çalışabilmesidir.

YAZILIM MÜHENDİSLİĞİ EKONOMİSİ

Yazılım mühendisliğinin önemli amaçlarından birisi, bir yazılım ürününün geliştirilmesine yönelik maliyetleri göz önüne almak ve uygun sınırlar içinde tutmaktır. Bir örnekle anlatmak gerekirse; yeni bir kodlama yönteminin piyasaya çıktığını varsayalım. Bu yeni çıkan kodlama yönteminin piyasada kullanılan mevcut kodlama yönteminden %15 daha hızlı olduğu yönünde söylem ortaya atılıyor. Bir yazılım mühendisi olarak çalıştığınız yazılım firmasındaki projelerinizi bu yeni kodlama yöntemine taşır mıydınız?

Bu soruya evet demeden önce; yeni kodlama yöntemi için gerekli eğitim maliyeti, yeni kodlama yönteminin paydaşlara tanıtımındaki olumlu veya olumsuz etkileri, mevcut yazılım projelerinin yeni kodlama yöntemi ile uyarlanması sonucu oluşacak bakım maliyetleri gibi koşulları dikkate alınması gerekir.

Diğer bir yandan yeni kodlama yöntemi, kullanılan mevcut kodlama yöntemine göre hangi açıdan %15 daha hızlıdır?

Yeni yöntem kodlama süresini mi %15 azaltıyor yoksa yazılım projesinin bakım maliyetini mi %15 azaltıyor?

Bir yazılım mühendisinin bu noktada ekonomik faktörleri göz önünde bulundurması gerekmektedir.

YAZILIM GELİŞTİRME SÜRECİ BİR EKİP İŞİ Mİ, YOKSA BİREYSEL BİR KOD YAZMA AKTİVİTESİ MİDİR?

Bu sorunun cevabı projenin büyüklüğüne göre değişecektir. Yazılım geliştirme süreci, bireysel bir kod yazma aktivitesi olarak başlayıp bir ekip işine dönüşen sistem oluşturma sürecidir. İyi bir yazılım ekibi olmak kısmen kolay olsa da yine de belli zorlukları vardır. Ekip üyelerinin almış oldukları eğitim ve teknik konulardaki bilgi düzeyleri arasında farklılıklar olabilir. Bunun yanında, yazılım projelerinin karmaşıklığı da önemli bir faktördür.

Başarılı bir yazılım ekibi oluşturabilmek için ekip üyeleri arasında etkin bir iletişim alt yapısının oluşturulması gerekir. Bu noktada takım lideri ve proje yöneticisine önemli görevler düşmektedir. Takım lideri, ekibin teknik anlamda yönetiminden proje yöneticisi, teknik olmayan bütün yönetimsel kararlardan sorumludur.

YAZILIM MÜHENDİSLİĞİ ETİĞİ

Diğer mühendislik disiplinleri gibi, yazılım mühendisliği, bu alanda çalışan kişilerin özgürlüğünü sınırlayan sosyal ve yasal bir çerçevede yürütülür. Bir yazılım mühendisi olarak, mesleğinizin sadece teknik becerilerin uygulanmasından daha geniş sorumluluklar içerdiğini kabul etmelisiniz. Yeteneklerinizi ve becerilerinizi dürüst olmayan bir şekilde veya yazılım mühendisliği mesleğine itibarsızlaştıracak şekilde kullanmamalısınız. Ancak kabul edilebilir davranışların standartlarının yasalara bağlı olmadığı fakat daha zayıf mesleki sorumluluk kavramına bağlı olduğu alanlar vardır. Bunların bazıları:

DURUM ÇALIŞMALARI

Yazılım mühendisliği kavramlarını örnekle açıklamak için dört farklı tür sistemden örnekler vermek gerekirse;

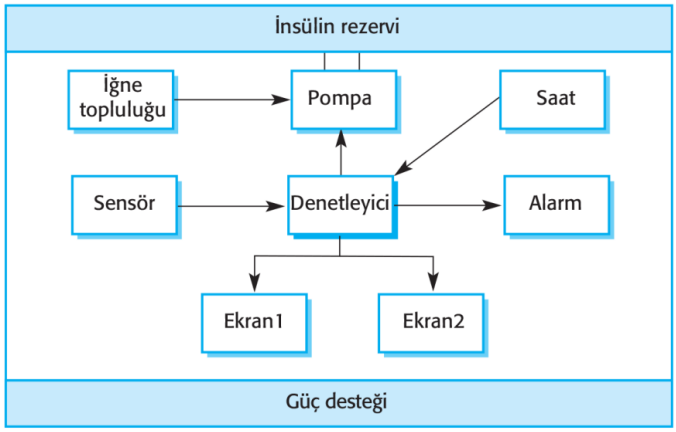
BİR İNSÜLİN POMPASI KONTROL SİSTEMİ (Gömülü Sistem)

İnsülin pompası pankreasın (bir iç organ) işleyişini taklit eden medikal bir sistemdir. Bu sistemi kontrol eden yazılım bir sensörden bilgi toplayan ve kullanıcıya kontrollü olarak insülin dozu veren bir pompayı yöneten bir gömülü sistemdir. Diyabetli kişiler bu sistemi kullanırlar.

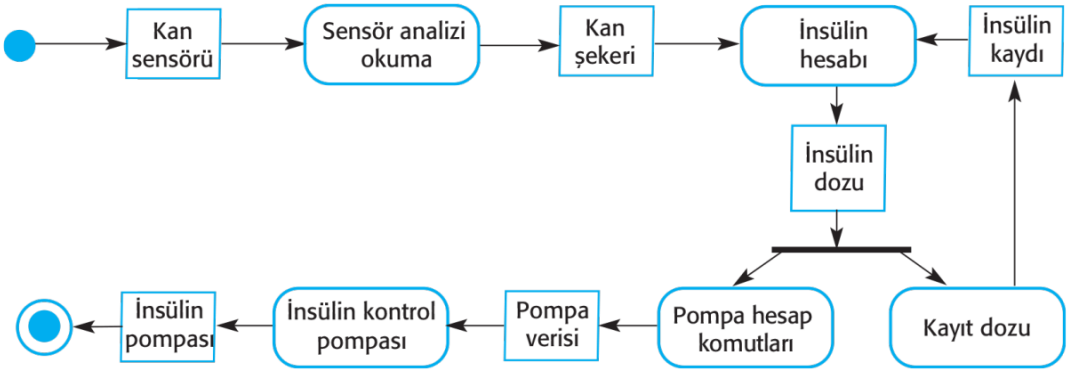
Diyabet, kişinin pankreasının insülin adı verilen hormonu yeterli miktarlarda üretemediği bir durumdur.

İnsülin kandaki glikozu (şeker) metabolizma için kullanır. Diyabetin geleneksel tedavisi genetik olarak üretilmiş insülinin düzenli enjeksiyonlarını içerir. Diyabetliler düzenli olarak kan şekeri düzeylerini dışsal bir ölçü kullanarak ölçerler ve enjekte etmeleri gereken insülin dozunu tahmin ederler.

İnsülin pompasının donanım mimarisi



İnsülin pompasının etkinlik diyagramı



RUH SAĞLIĞI İÇİN BİR HASTA BİLGİ SİSTEMİ (Bilgi Sistemi)

Ruh sağlığını desteklemek için bir hasta bilgi sitemi (ZihinSaS sistemi), ruh sağlığı problemleri olan hastalar ve aldıkları tedaviler hakkında bilgi tutan bir tıbbi bilgi sistemidir. ZihinSaS sistemi kliniklerde kullanımı amaçlanmış bir hasta bilgi sistemidir.

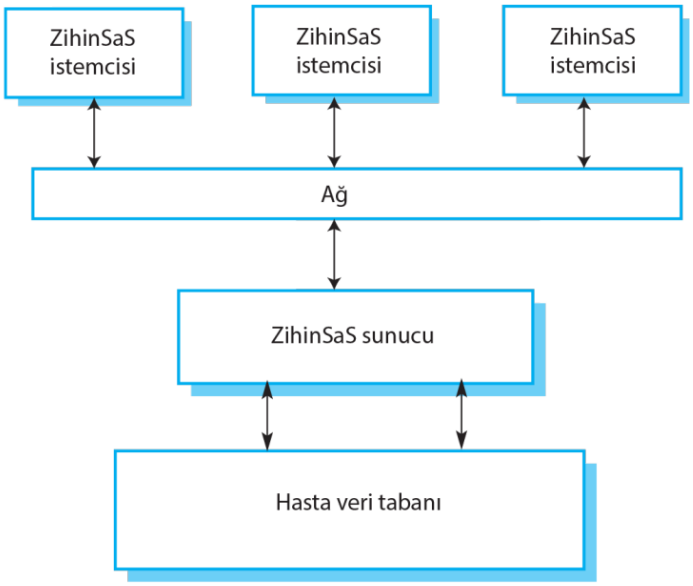
Merkezileştirilmiş bir hasta bilgi veri tabanı kullanır fakat bir dizüstü bilgisayarda çalışabilecek şekilde tasarlanmıştır, bu nedenle güvenli ağ bağlantısı bulunmayan sitelerden erişilip kullanılabilir.

Sistem tam bir tıbbi kayıt sistemi değildir ve bu nedenle diğer tıbbi koşullar hakkında bilgi tutmaz. Fakat diğer klinik bilgi sistemleriyle etkileşebilir ve veri değişimi yapabilir.

Sistemin iki amacı vardır:

- Sağlık servis yöneticilerinin yerel ve devlet hedeflerine göre performansının değerlendirebilmesi için yönetim bilgisi oluşturmak.
- Hastaların tedavisini desteklemek için tıbbi personele zamanında veri sağlamak.

ZihinSaS sistem organizasyonu



OKULLAR İÇİN SAYISAL BİR ÖĞRENME ORTAMI (Destek Ortamı)

Birçok öğretmen eğitimi desteklemek için etkileşimli yazılım sistemlerinin kullanılmasının hem artırılmış öğrenci motivasyonu hem de öğrencilerde daha derin bir bilgi düzeyi ve anlayış oluşturabileceğini iddia ederler. Fakat bilgisayar destekli öğrenme için en iyi strateji konusunda genel bir anlaşma yoktur ve öğretmenler öğrenmeyi desteklemek için pratikte bir çok farklı etkileşimli, web tabanlı araç kullanırlar.

Sistem, tüm sistem bileşenlerinin değiştirilebilir servisler olarak düşünüldüğü bir servis tabanlı sistemdir.

Sistemde üç tür servis vardır:

1. Yardımcı servisler,
2. Uygulama servisleri,
3. Konfigürasyon servisleri.

Sayısal bir öğrenme ortamının (eogren) mimarisi

KAYNAKLAR

Bilişimde Sistem Analizi ve Tasarımı, Oya H. YÜREGİR, Nobel Kitabevi , ISBN: 975-85-61-05-7

Sommerville, I. (2011). Software engineering (ed.). *America: Pearson Education Inc.*

Yücalar, F. ve Borandağ, E. (2022). Yazılım Mühendisliğinde Modern Yaklaşımlar