Terim	İngilizce	Açıklaması	İlgili Yazı
Açık mimari	Open Architecture	Açık Mimari, açık standartları kullanarak geliştirilmiş ve yeni bileşen ekleme, değiştirme ve yerine farklısını koyma yada çıkartmayı kolaylaştırmış bilgisayar yada yazılım mimarisi olarak tanımlanabilir. Açık ve ortak kabul görmüş standartların kullanımı Arayüz protokol ve veri standartlarının kabul edilmesi Standart servislerin belirlenmesi Birden fazla üretici tarafından sağlanan ürünlerin kullanımı En az çaba ile birlikte çalışabilirlik Kolayca ölçeklenebilir ve güncellenebilir sistemler Taşınabilir uygulamalar Taşınabilir kullanıcılar(kullanıcıların aynı amaçlı farklı sistemlere kolayca adapte olması)	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/a%C3%A7%C4%B1k-sistem-mimarileri-1-temel-%C3%B6zellikler-6a413b754a62
Arakatman	middleware	Dağıtık ortamda farklı çalışabilir modüller olarak geliştirilen yazılımın bir bütün olarak çalışabilmesi için başta veri alışverişi olmak üzere farklı seviyelerde işbirliği yapmaları gerekmektedir. Dağıtık ortamda çalışan bu uygulamaların iletişimi için kendimizin geliştireceği TCP/IP yada UDP protokolü ile haberleşen kütüphanelerden yerine Arakatman yazılımları dağıtık uygulamalar ile işletim sistemi ve haberleşme protokolü arasında bir katman olarak görülmektedir. Heterojen bir sistemde tüm farklılıkları yöneterek üst seviye uygulamaları karmaşıklıktan kurtarır. Aynı zamanda bu uygulamaların herhangi bir anda hangi bilgisayar üzerinde çalıştığı, o an hayatta olup olmadığı, veriyi almaya hazır olup olmadığı gibi daha birçok teknik problemi çözmek üzere tasarlanmaktadır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/arakatmanlar-1-arakatman-yaz%C4%B1l%C4%B1mlar%C4%B1-genel-bilgiler-c6c8d5e8f23a
Bağımlılık	Coupling	Bağımlılık kod seviyesi (kod), kurulabilme (deployment) veya çalışma zamanı (execution time) olabilmektedir. Her bir seviye bağımlılığın azaltılması farklı açılardan projeye fayda sağlayacaktır. Veri Yapısı Bağımlılığında bir bileşen diğer bir bileşen ile bir nesne yada veri yapısının tamamını paylaşır. Kontrol bağımlılığında bir başka bir bileşenin nasıl işleyeceği ile ilgili kontrol algoritmasını belirler. Dış bağımlılıkta yazılım bileşeni dış bir donanıma yada başka bir yazılıma bağımlıdır. Ortak bağımlılıkta ise iki bileşen ortak bir veri yapısına bağımlıdır. İçerik Bağımlılığında bir bileşen diğer bileşenin verisini yada kontrol akışını değiştirir. Bu en kötü bağımlılıktır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/tasar%C4%B1m%C4%B1n-temelleri-2-ba%C4%9F%C4%B1ml%C4%B1l%C4%B1k-ve-uyumluluk-tasar%C4%B1m-prensipleri-20e28490e985
Bileşen tabanlı mimari	Component Based Architecture	Sistem küçük bileşenler ve bu bileşenleri bir arada tutan bileşen yöneticisinden meydana gelir. Bileşen yöneticisi çalışma zamanında kendisine tanımlanan bileşenleri başlatabilir, durdurabilir. Bileşenlerin genelde tanımlı standart ara yüzleri bulunmaktadır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimari-tasar%C4%B1m-kal%C4%B1plar%C4%B1-94a4f47d32a8

Desen	Pattern	Mimari desenler ise tekrarlanan tasarım problemlerine bulunmuş ve çalıştığı ispat edilmiş çözüm yöntemleridir. Bina mimari tasarımında kullanılan tasarım kalıpları ile benzer yaklaşımdadır. Mimari desenler, yazılım geliştirmede kullanılan tasarım desenlerine göre daha üst seviyede görülmektedir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac
Durum Makinesi	State Machine	Nesnelerin kontrol ve durum akışını modellemek için kullanılır. Durum(State) tasarladığımız sistemin belirli bir işi yaptığı yada başka bir duruma geçmek için uyarı beklediği koşullardır. laylar (Event) sistemde durum değişikliğine sebep olan iç veya dış itkilerdir. Bir tuşa basılması, belirli bir zamanın geçmesi, kapının açılması birer olay olarak durum değişikliğine sebep olurlar.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/g%C3%B6m%C3%BCl%C3%BC-ve-ger%C3%A7ek-zamanl%C4%B1-tasar%C4%B1m-kal%C4%B1plar%C4%B1-1-durum-makineleri-e9ab5200decb
Kalite Öznitelikleri Çalıştayı	Quality Attribute Workshop	Proje paydaşlarını bir araya getirerek etkileşim ile gereksinimler üzerinde hemfikir olmayı hedefler. Bu etkileşim geliştirilen sistemin paydaşların beklentileri ile uyumlu olmasını ve projenin geliştirme aşamasında başarısızlığa sebep olacak hataya dayanıklılık, performans gibi kalite gereksinimlerinin erken aşamada tasarıma girmesini sağlayacaktır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/kalite-%C3%B6znitelikleri-%C3%A7al%C4%B1%C5%9Ftay%C4%B1-ya-da-kalite-fikir-f%C4%B1rt%C4%B1nas%C4%B1-a521409c7a23
Kalite Öznitelikleri Tabanlı Tasa	Attribute Driven Design-ADD	Kalite Öznitelikleri Tabanlı Tasarım öncelikleana kullanım durumları, kalite öznitelik senaryoları, mimari kaygılar ve proje sınırlamaları girdilerin değerlendirilmesi ile başlıyor. Bu veriler değerlendirildikten sonra tekrarlı bir şekilde yazılım mimarisi tasarımı üst seviyeden başlanılarak detaylandırılır. Tasarım problemi daha alt problemlere bölünür. Bu döngüde hangi problemi adresleyeceğine karar verirsin. Genelde ilk iterasyonda sistem mimarisinin ana iskeleti oluşturulur. Daha sonra bu iskeleti destekleyecek ek yapılar tasarlanır. İhtiyaç duyulması durumunda üçüncü ve devamındaki döngülerde hala çözüm üretilmemiş problemler çözümlenir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarileri-5-kalite-%C3%B6znitelikleri-tabanl%C4%B1-tasar%C4%B1m-fb38d79376f3
Katmanlı Mimari	Layered Architecture	Her katman farklı bir fonksiyonu yerine getirir ve genel olarak kendisinden bir alt ve bir üst katman ile iletişim kurar. Katmanlı mimari kalıbında eğer bir katman sadece bir alt katmandan servis alabiliyor ise bu tür sistemlere kapalı katmanlı sistem diyoruz. Bir katman kendinden iki ve daha fazla altta olan katmanlara da erişiyor ise buna açık katmanlı sistem denmektedir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimari-tasar%C4%B1m-kal%C4%B1plar%C4%B1-94a4f47d32a8
Mevcut Sistemde Tasarım Değ	BrownField System	Mevcut bir sistemin üzerine sistemin geçmişten gelen tasarımsal kısıtlarını göz önünde tutarak ekleme ve değişiklikler yapılır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarileri-5-kalite-%C3%B6znitelikleri-tabanl%C4%B1-tasar%C4%B1m-fb38d79376f3

Mikro-Servis Mimarisi	Microserivce Architecture	Mikro servis kavramındaki temel bakış servislerin mikro olmasından çok anlamlı ve tek bir iş mantığımı sağlıyor olmasıdır. Aynı zamanda yönetilebilir küçük bir takım tarafından geliştirilebilir olması önem verilen bir noktadır. Mikro servis mimarisi aslında dağıtık mimarinin temel faydalarından faydalanmak üzere gelişmiştir. Servisler birbirinden bağımsız şekilde yüklenebilir. Bir servis istenmeden kapandığında diğer servisleri etkilemez. Kritik servisler birden fazla kopya çalıştırılarak genişleyebilen yapı sağlanır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimari-tasar%C4%B1m-kal%C4%B1plar%C4%B1-94a4f47d32a8
Mimari Erozyon	Architectural Erosion	projenin ileri ki aşamalarında takvim sıkışıklığı, yazılımcı değişikliği gibi faktörler ile mimari kararlar doğru şekilde yazılıma aktarılmaz yada mevcut bir özellik istenmeden bozulur.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimarinin-evrimi-3-yeniden-d%C3%BCzenleme-yada-yeniden-yazma-a54205e0ac19
Mimari Kaygılar	Architectural Concerns	Mimari kaygılar ise alınması gereken tasarım kararlarını ifade eder. Örnek olarak alınan verinin doğrulaması, iletişim mekanizmaları, verinin yedeğinin alınması ve hata yönetimi sayılabilir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-2-mimari-gereksinimler-bf010e41f99c
Mimari Ödün Analizi Metodu	(Architectural Tradeoff Analysi	ATAM incelenen mimarinin potansiyel risklerini, ödünleşim ve duyarlılık noktalarını tespit eden bir yazılım mimarisi değerlendirme ve analiz etme yöntemidir. ATAM, alınan tasarım kararlarının sonuçlarını sistemden beklenen kalite öznitelikleri doğrultusunda değerlendirmeyi amaçlar.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-9-mimarinin-de%C4%9Ferlendirilmesi-8c0657c514b7
Olay Fırtınası	Event Stroming	İş alanını daha iyi analiz edilerek fonksiyonel gereksinimlerin oluşturulması hedefleyen bir yöntemdir. Bu yaklaşım son zamanlarda (İş alanı Tabanlı Tasarım (Domain Driven Design) için etkin olarak kullanılmaktadır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/kalite-%C3%B6znitelikleri-%C3%A7al%C4%B1%C5%9Ftay%C4%B1-ya-da-kalite-fikir-f%C4%B1rt%C4%B1nas%C4%B1-a521409c7a23
Oyun fırtınası	Game-Storming	Bir konuda ilgili bütün paydaşların bir araya gelerek problemi yaratıcı bir şekilde oyun yöntemi ile analiz edip çözüm üretmeyi hedefler. Yaratıcılığı ve paydaşlar arası sahiplenmeyi arttırdığı için son yıllarda aktif olarak tercih edilmektedir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/kalite-%C3%B6znitelikleri-%C3%A7al%C4%B1%C5%9Ftay%C4%B1-ya-da-kalite-fikir-f%C4%B1rt%C4%B1nas%C4%B1-a521409c7a23
Paydaş	Stakeholder	Yazılım mimarisinin paydaşları mimari ile ilgisi olan geliştirici, test mühendisi, kullanıcı, üst yönetim gibi kişilerden oluşur. Bir yazılım için paydaşları belirler iken, “yazılımdan kimler etkilenecek” ve “kimler yazılımı etkileyecek” soruları sorulur.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac
Performans	Performance	Yazılımın zaman ile ilgili ilgili gereksinimleri sağlama özelliğidir. Bir olay olduğunda sistemin tamamı yada bir bölümü bu olaya belirlenen zamanda cevap vermesi beklenir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-notlar%C4%B1-4-performans-kalite-%C3%B6zniteli%C4%9Fi-2c7345d924e3
Referans mimariler	Reference Architectures	Referans mimariler ise belirli bir alandaki yazılım mimarisinin ihtiyaçlarına karşılık veren ve genelde birden fazla tasarım kalıbını içeren bütünsel bir mimaridir. Belirli bir referans mimariyi temel alan yazılımcı referans mimaride tanımlı katmanlar içerisinde referans mimaride tanımlı bileşenleri ve bu bileşenler arasında ise yine tanımlı ilişkileri kurar.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac

Servis Tabanlı Mimari	Service Oriented Architecture	farklı kabiliyetler servis olarak geliştirilmekte ve istemcilerin istekleri bu servislere iletildiğinde ilgili servis diğer servislerden tanımlı ara yüzler üzerinden hizmet alarak istemcinin ihtiyacını karşılamaktadır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimari-tasar%C4%B1m-kal%C4%B1plar%C4%B1-94a4f47d32a8
Sıfırdan Tasarım	Greenfield System	Yazılım tasarımı tamamen yeni bir alanda yapılıyor ise mimari tasarım çok daha yaratıcı ve zorlayıcı olabilmektedir. Bu tür sistemlerin tasarımına sıfırdan tasarım denmektedir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarileri-5-kalite-%C3%B6znitelikleri-tabanl%C4%B1-tasar%C4%B1m-fb38d79376f3
Süreklilik	Availability	Süreklilik kalite özniteliği bir sistemin ihtiyaç duyulduğunda kullanılabilir ve faydalanılabilir olma derecesidir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-notlar%C4%B1-3-s%C3%BCreklilik-kalite-%C3%B6zniteli%C4%9Fi-c3566777e66c
Taktikler	Tactics	Mimari taktikler temel seviyedeki çözüm tekniklerdir. Yazılım geliştiriciler bu temel seviyedeki taktikleri yıllardır kullanmaktadır. Örneğin yazılımın performansını arttırmak için verinin birden fazla kopyasını tutma, kaynakları arttırma, koşut çalışmayı sağlama kullanılabilecek taktikler arasındadır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac
Teknik Borç	Technical Debt	İdeal olmayan kod, tasarım yada mimari, bir problemi çözmek için kullandığınız kestirme yol veya dikkate almadığımız hata veya operasyonel durumların sonradan tespit edilip kodda düzeltilmesi ekstra maliyet getirecektir. Bu durum finansal sistemden esinlenerek teknik borç olarak nitelendirilir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimarinin-evrimi-1-teknik-bor%C3%A7-nedir-e87d89fe12de
Teslim Zamanı	Deadline	Bir işin bitmesi gereken en son zamanı belirtir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-notlar%C4%B1-4-performans-kalite-%C3%B6zniteli%C4%9Fi-2c7345d924e3
Test Edilebilir Tasarım	Design for Testability	Tasarımın içine test ile ilgili özelliklerin eklenmesidir.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarileri-8-test-edilebilir-tasar%C4%B1m-ff558effa027
Test edilebilirlik	Testability	Test edilebilirlik kalite özniteliği bir yazılımın hatalarının test ile ne kadar kolay yakalanabileceği ölçüsüdür. Test Edilebilirlik kalite özniteliğinin öncelikle gözlenebilirlik, kontrol edilebilirlik ve boyutsal küçüklük kalite özniteliklerinin ile alakalıdır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarileri-8-test-edilebilir-tasar%C4%B1m-ff558effa027
Uyumluluk	Cohession	Bileşen içi uyumluluk ile ilgili sınıflandırma aşağıdaki gibidir: Tesadüfi Uyumluluk alakasız elementlerin aynı bileşen içinde yer almasıdır. Mantıksal uyumluluk mantıksal olarak benzer fakat yöntemsel olarak farklı işlemlerin bir arada bulunmasıdır. Zamansal uyumlulukta aynı zaman içinde/ sırasında yapılacak işler aynı bileşen içinde yer alır. Yordam Uyumluluğu işleme sırası peş peşe olan işlemler aynı bileşen içinde yer alır. Haberleşme uyumluluğunda aynı girdi üzerine işlem yapan elemanlar aynı bileşende yer alır. Fonksiyonel uyumlulukta tek bir fonksiyonu yerine getiren işlemler aynı bileşen içinde yer almaktadır. Bu ideal durumdur.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/tasar%C4%B1m%C4%B1n-temelleri-2-ba%C4%9F%C4%B1ml%C4%B1l%C4%B1k-ve-uyumluluk-tasar%C4%B1m-prensipleri-20e28490e985

Yazılım Mimarileri	Software Architecture	"	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac
Yeniden düzenleme	Refactor	Yeniden düzenleme yazılımın iç yapısının düzeltilirken dış davranışının aynı kalması olarak tanımlanmaktadır. Mimari yeniden düzenlemede ise amaç sistemin değiştirilebilirlik, performans, test edilebilirlik, değiştire bilirlik,bakım yapılabilirlik ve süreklilik gibi kalite öz niteliklerini iyileştirmek olacaktır.	https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/mimarinin-evrimi-2-teknik-bor%C3%A7-nas%C4%B1l-%C3%B6denir-35f993a7e95c