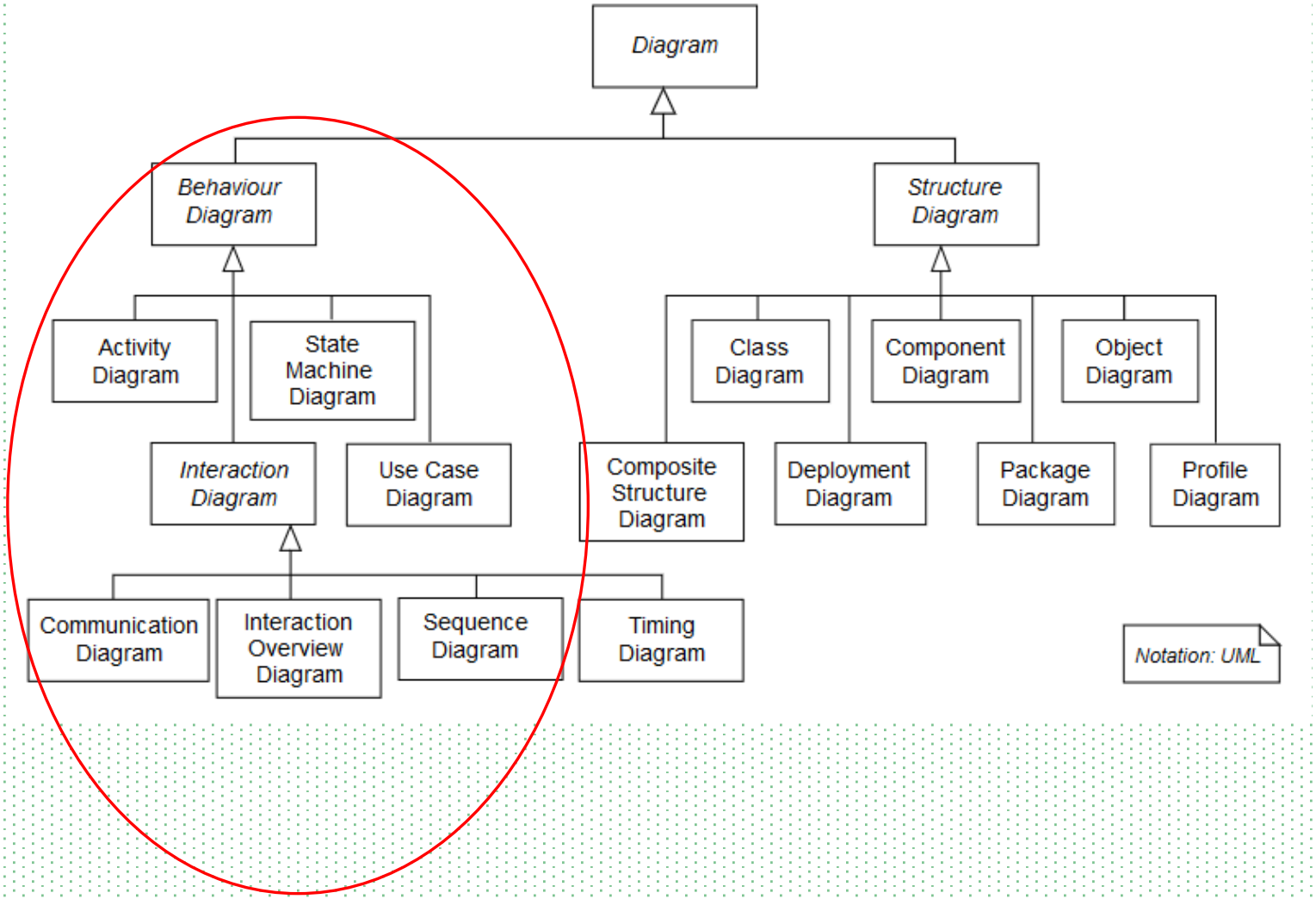
YMH 311-YAZILIM TASARIM VE MİMARİSİ

Dr. Öğr. Üyesi BİHTER DAŞ

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

6. HAFTA:
DAVRANIŞ DİYAGRAMLARI

UML Modelleme



1. Kullanım senaryosu (Use case) Diyagramları

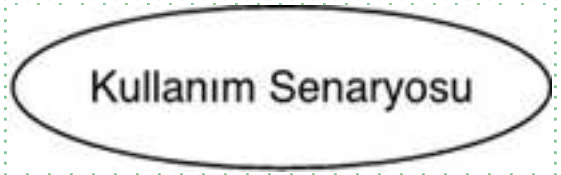
- Bir kullanıcı ve bir sistem arasındaki etkileşimi anlatan senaryo topluluğudur.
- Bir sistemdeki farklı rol türlerini ve bu rollerin sistemle nasıl etkileşime girdiğini görselleştirmenizi sağlar. (Temel kullanım amacı)
- Sistemin üst düzey bir görünümü için kullanılır. Özellikle yöneticilere veya paydaşlara sunum yaparken kullanışlıdır. Sistemin iç işleyişine derinlemesine inmeden sistemle etkileşime giren rolleri ve sistemin sağladığı işlevselliği vurgulayabilirsiniz.
- Dahili ve harici faktörleri belirlemek için kullanılabilir. Büyük karmaşık projelerde bir sistem, başka bir kullanım durumunda harici bir rol olarak tanımlanabilir.

Use case diyagramları aşağıdaki nesneden oluşur.

- Aktör

Kullanım durumu diyagramındaki aktör, belirli bir sistemde rol oynayan herhangi bir varlıktır. Bu bir kişi, kuruluş veya harici bir sistem olabilir

- Kullanım senaryosu

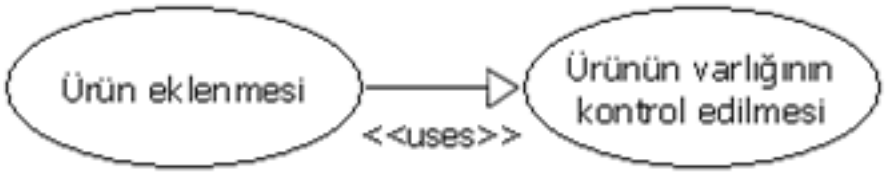


Sistem içindeki bir işlevi veya eylemi temsil eder. Oval olarak çizilir ve fonksiyon ile isimlendirilir.

- İlişkiler

Bu modelde senaryolar arasında **içerme (inclusion)** ve **genişleme (extension)** olmak üzere iki tür ilişki kurulabilir. İçerme ilişkisi bir ana senaryonun bir alt senaryo içerdiğini belirtir. Söz konusu alt senaryo birden fazla senaryo tarafından kullanılır. Bu senaryolar arasında noktalı bir çizgi çizilir ve üzerine <<include>> ifadesi yazılır. Genişleme ilişkisi mevcut senaryoya yeni bir senaryo eklemek için kullanılır. Bu ilişkiyi belirtmek için noktalı çizginin üzerinde <<extends>> ifadesi yazılır.

İçerme (Include, uses): İçerme ilişkisi senaryo(lar) içerisinde iki veya daha fazla tekrarlanan adım var ise tekrar edilen kısımlar ayrı Use case olarak bölüp içerme ilişkisi kullanılmalıdır. Bir senaryonun içinden bir alt programa dallanıp geri dönmek gibidir.



Genişletme (Extend): Senaryo grupları doğal akışa göre hazırlanır. Çeşitli koşullar altında bu doğal akıştan sapmalar olabilir. Genişletme ilişkisi ana senaryodan ayrılma noktasından sonra yapılanları belirtir.

Genelleme (Generalization): Sınıflar arasındaki türeme ilişkisine benzer. Genel bir senaryo grubundan özel bir senaryo grubu türetilir.

Use Case kullanımının avantajları

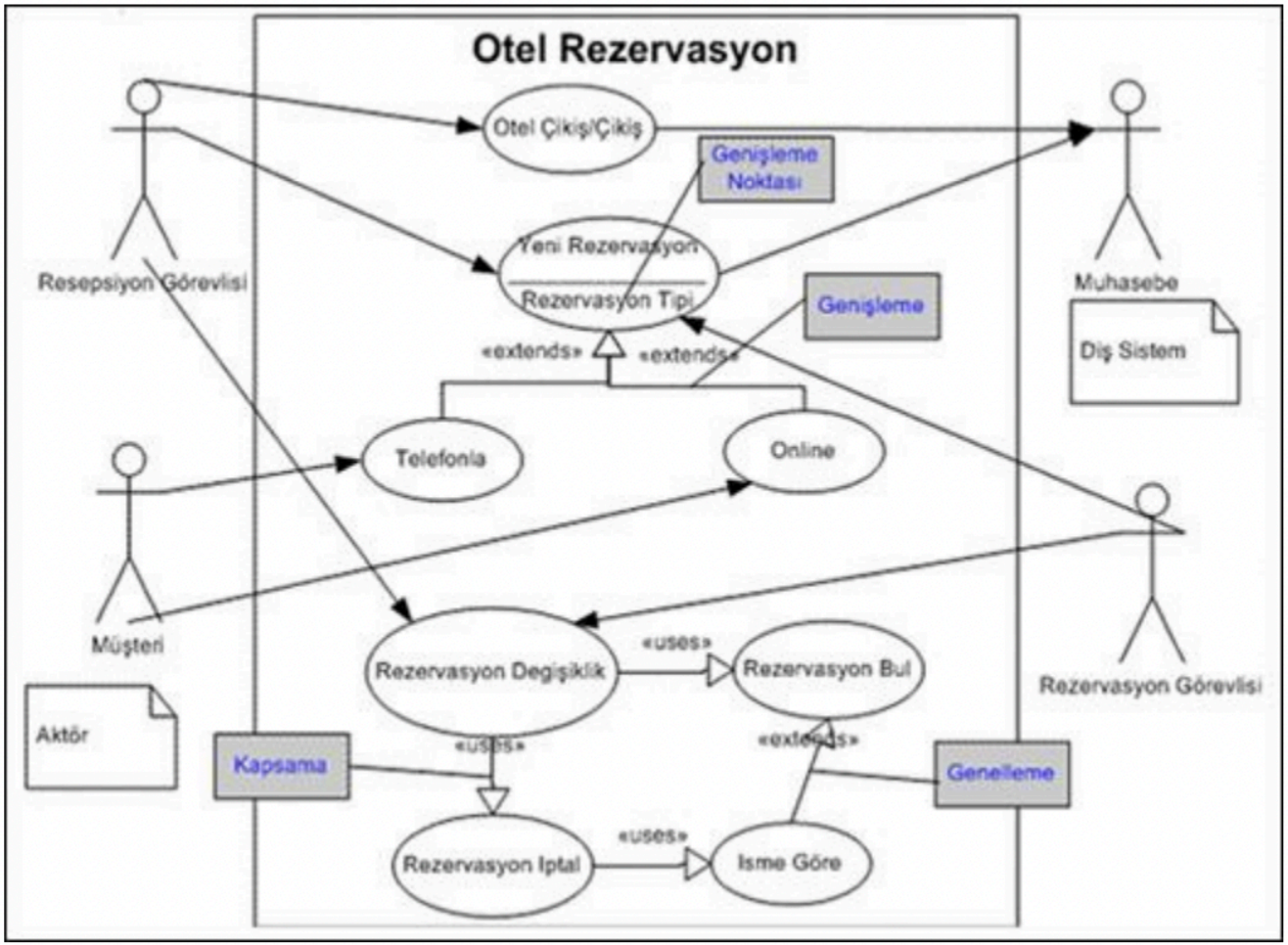
Sistemin erişimini, sınırlarını belirler. Böylelikle geliştirilecek sistemin boyutunu ve karmaşıklığını kafamızda daha rahat canlandırabiliriz.

- Kullanım senaryoları isteklerin çözümlenmesine çok benzemektedir: daha nettir ve tamdır.
- Basit oluşu müşteri ile geliştirme ekibi arasında iletişime olanak tanır. Geliştirme aşaması için temel oluşturur.
- Sistem testi için temel oluşturur.
- Kullanıcı kılavuzu hazırlamaya yardımcı olur.

Use case diyagramında dikkat edilmesi gereken hususlar

- Aktörler tespit edilirken sistemde çok sık adı geçen özneler kullanılmalı Özneler çıkarılırken rollere göre gruplandırılmalıdır. Aktör sayısının en aza indirilmesine özen gösterilmelidir.
- Use Case diyagramları oluştururken cümleler arasında geçen kilit fiiller önemlidir. fiillerin çok sık geçmesi ve bunların projenin ana hedeflerine uygun, sisteme değer katan iş süreçlerini içermesine dikkat edilmelidir.
- Use Case diyagramlardaki en önemli ilişkinin içerme ilişkisidir. Zorunlu olmadıkça diğer ilişkileri fazla kullanmaktan kaçınılmalıdır.
- Use Case diyagramlarının kullanım amacını unutmamak gerekir. Çok fazla Use Case diyagramı çizmekten ziyade kolaylıkla anlaşılabilir, kullanıcılar için değer taşıyabilen çekirdek iş süreçlerini anlatan diyagramlar olmalıdır.

*** UML diyagramlarında bir şeklin anlamını açıklayan özel sözcükler (sterotype) <> simgeleri arasına yazılır. Aktörler çizgi adam şeklinde gösterildiği gibi bir dikdörtgen ile de ifade edilebilir. Bu durumda dikdörtgenin anlamını belirtmek için "streotype" kullanılır.



Şekil 5. Otel Rezervasyon Use Case Diyagramı

Örnek Diyagram ve Dökümantasyon

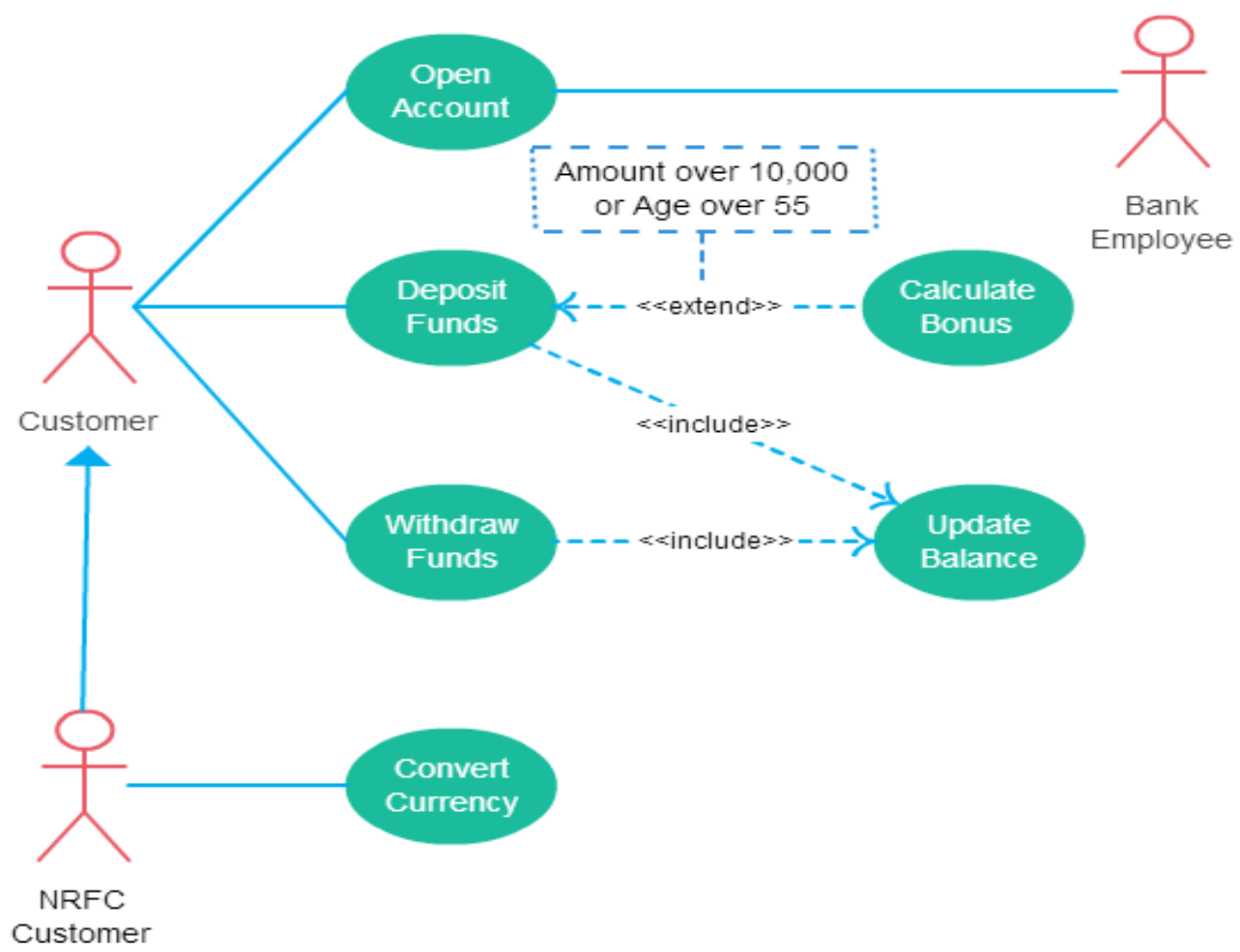
Otel rezervasyon sisteminde, sistemle etkileşimde olan 4 ana aktör ya da rol bulunmaktadır. Muhasebe aktörü genel bir kullanıcı olmayıp, otel rezervasyon sisteminden etkilenen dış sisteme ait bir yazılımı temsil etmektedir. Bunu aktörlerin Use Case'leri kullanırken çizilen bağlantıların ok yönünden kestirebiliriz. Use Case'lerden etkilenen aktörlerde ok yönü Use Case'den aktöre doğru iken normal kullanıcılarda, aktörden Use Case'lere doğru bir geçiş olduğunu siz de fark etmişsinizdir. Diyagramdan da görüldüğü gibi otel rezervasyon sistemi Otel Giriş/Çıkış, Yeni Rezervasyon, Rezervasyon Değişilik gibi üç ana Use Case'den oluşmakta, diğer Use Case'ler ise ana Use Case'ler ile ilişkide bulunan alt Use Case'ler olarak adlandırılabilir

Use Case Dökümantasyonu

Use case dökümantasyonunun standart bir formatı yoktur. Her firma kendisine uygun format belirleyebilir. Temel dökümantasyon formatı aşağıdaki gibidir. İhtiyaçlar doğrultusunda eklemeler yapılabilir.

- Adı: Yeni Rezervasyon
- Aktörler: Resepsiyon Görevlisi, Rezervasyon Görevlisi, Muhasebe
- Ön Koşullar: Sisteme login olunmalıdır Son Durum: Muhasebe sistemi ve rezervasyon kayıt sistemi güncellenir.
- Genişleme Noktası: Rezervasyon Tipi, müşteri bilgi kaydı
- Başarılı Senaryo:
    1. Müşteriye oda tipleri sunulur
    2. Müşteri oda tipini seçer
    3. Müşteri konaklama başlangıç tarihini ve konaklama süresini girer
    4. Müşteri mevcut odalar üzerinde arama yapar
    5. Eğer oda bulunamamışsa (A1)
    6. Oda bulunmuştur, fiyat görüntülenir
    7. Müşteri teklifi reddederse (A2)
    8. Müşteri bilgilerini girer
    9. Rezervasyon Yapılır
    10. Rezervasyonda anlaşılan fiyat kayıt edilir ve rezervasyon numarası verilir
    11. Muhasebe sistemi haberdar edilir
- Alternatif Yollar:
    - A1. farklı bir oda tipi yada tarih deneyin mesajı ile 2. adıma geri döner
    - A2. Use case iptal edilir ve herhangi bir değişiklik yapılmaz

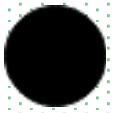
Diğer bir Use Case Örneği



2. Faaliyet diyagramı (Activity Diagram)

- Bir sistemdeki kontrol akışını göstermek için Aktivite Diyagramları kullanılır. Aktivite diyagramlarını kullanarak sıralı ve eşzamanlı aktiviteleri modelliyoruz. Bu nedenle, temel olarak bir aktivite diyagramı kullanarak iş akışlarını görsel olarak tasvir ediyoruz.
- Bir aktivite diyagramı, akışın durumuna ve gerçekleştiği sıraya odaklanır. Bir etkinlik diyagramı kullanarak belirli bir olaya neyin neden olduğunu açıklar veya tasvir ederiz.
- Aktivite diyagramı, akış şemasına çok benzer. Programcı olmayanlar, iş akışlarını modellemek için akış çizelgelerini kullanır. Örneğin: Bir üretici, belirli bir ürünün nasıl üretildiğini açıklamak ve göstermek için bir akış şeması kullanır. Bir akış şemasına, bir aktivite diyagramının ilkel bir versiyonu diyebiliriz.

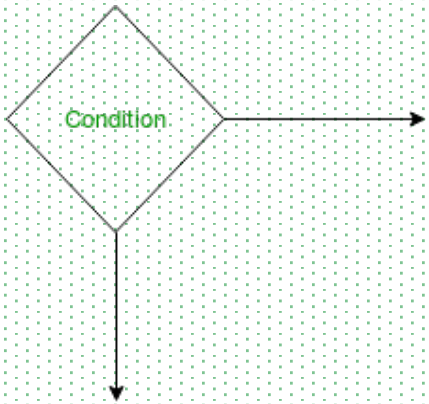
Activity diyagram gösterimleri



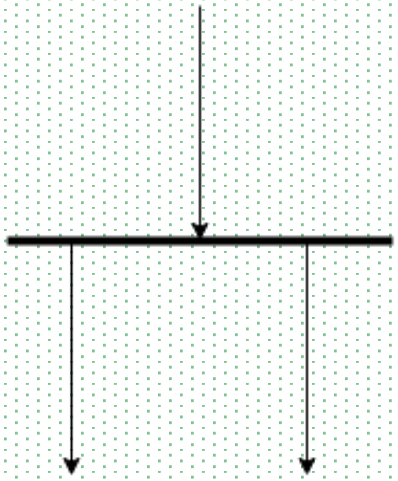
Başlangıç durumu

Köşeleri yuvarlatılmış bir dikdörtgen kullanarak bir etkinliği temsil ediyoruz. Temel olarak gerçekleşen herhangi bir eylem veya olay, bir aktivite kullanılarak temsil edilir.

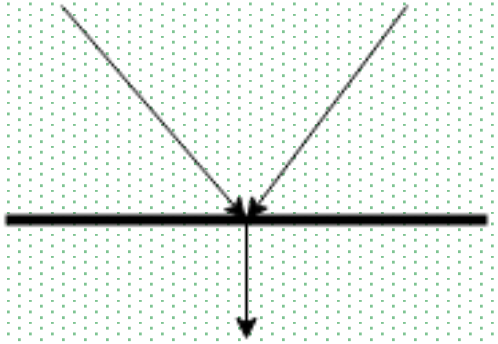
Bir aktivite durumundan diğerine geçişi göstermek için kullanılırlar.



Kontrol akışına karar vermeden önce bir karar vermemiz gerektiğinde karar düğümünü kullanırız.

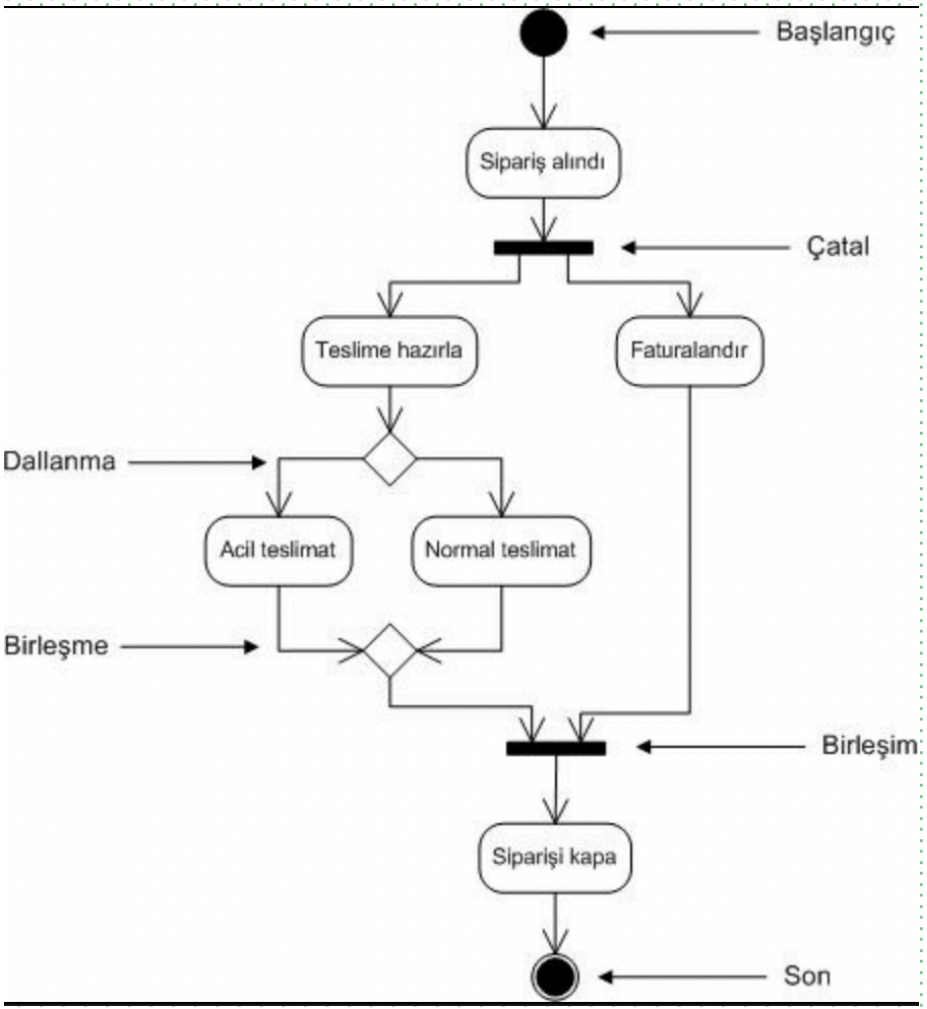


Her iki aktivite aynı anda yürütüldüğünde, çatal kullanılır.



Birleştirme düğümleri, birbirine yakınlaşan eşzamanlı etkinlikleri desteklemek için kullanılır. Birleştirme notasyonları için iki veya daha fazla gelen kenarımız ve bir giden kenarımız var.

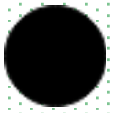
Activity diyagram UML örneği



3. Durum diyagramı (State machine)

- Sonlu zaman örneklerinde sistemin veya sistemin bir bölümünün durumunu temsil etmek için bir durum diyagramı kullanılır.
- Bu bir davranış diyagramıdır ve sonlu durum geçişlerini kullanan davranışı temsil eder. Durum diyagramları ayrıca Durum makineleri ve Durum diyagramları olarak da adlandırılır. Bu terimler genellikle birbirinin yerine kullanılır.
- Zaman içinde sistemdeki dinamik değişimi modellemek için kullanılır.
- Durum diyagramının temel amacı, değişikliklere neden olan süreçleri veya komutları değil, sınıfın durumundaki çeşitli değişiklikleri tasvir etmektir.

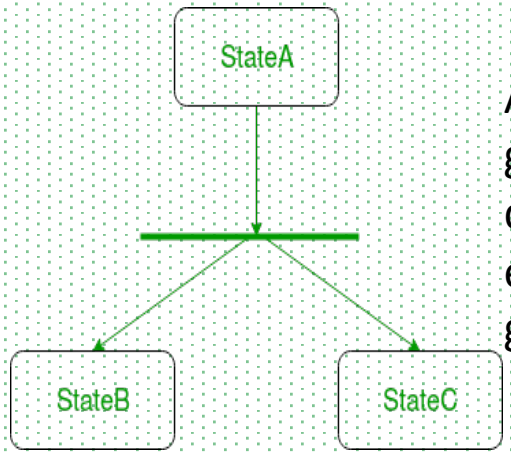
Durum diyagram gösterimleri



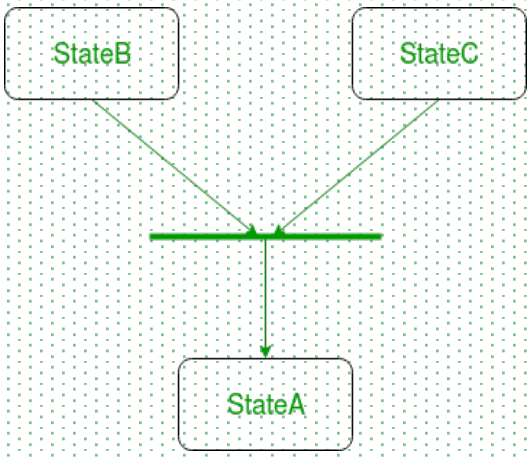
Bir sistemin yada sınıfın başlangıç durumu

Geçiş – Kontrolün bir durumdan diğerine geçişini veya değişimini temsil etmek için düz bir ok kullanırız.

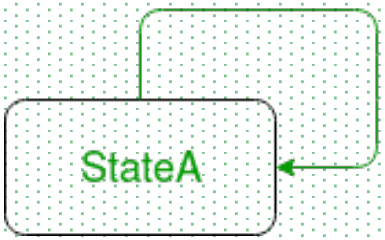
Bir durumu temsil etmek için yuvarlak bir dikdörtgen kullanırız.



Ana durumdan gelen ok ve yeni oluşturulan durumlara doğru giden oklarla bir çatal gösterimini temsil etmek için yuvarlak, içi dolu bir dikdörtgen çubuk kullanırız. İki veya daha fazla eşzamanlı duruma bölünen bir durumu temsil etmek için çatal gösterimini kullanırız.

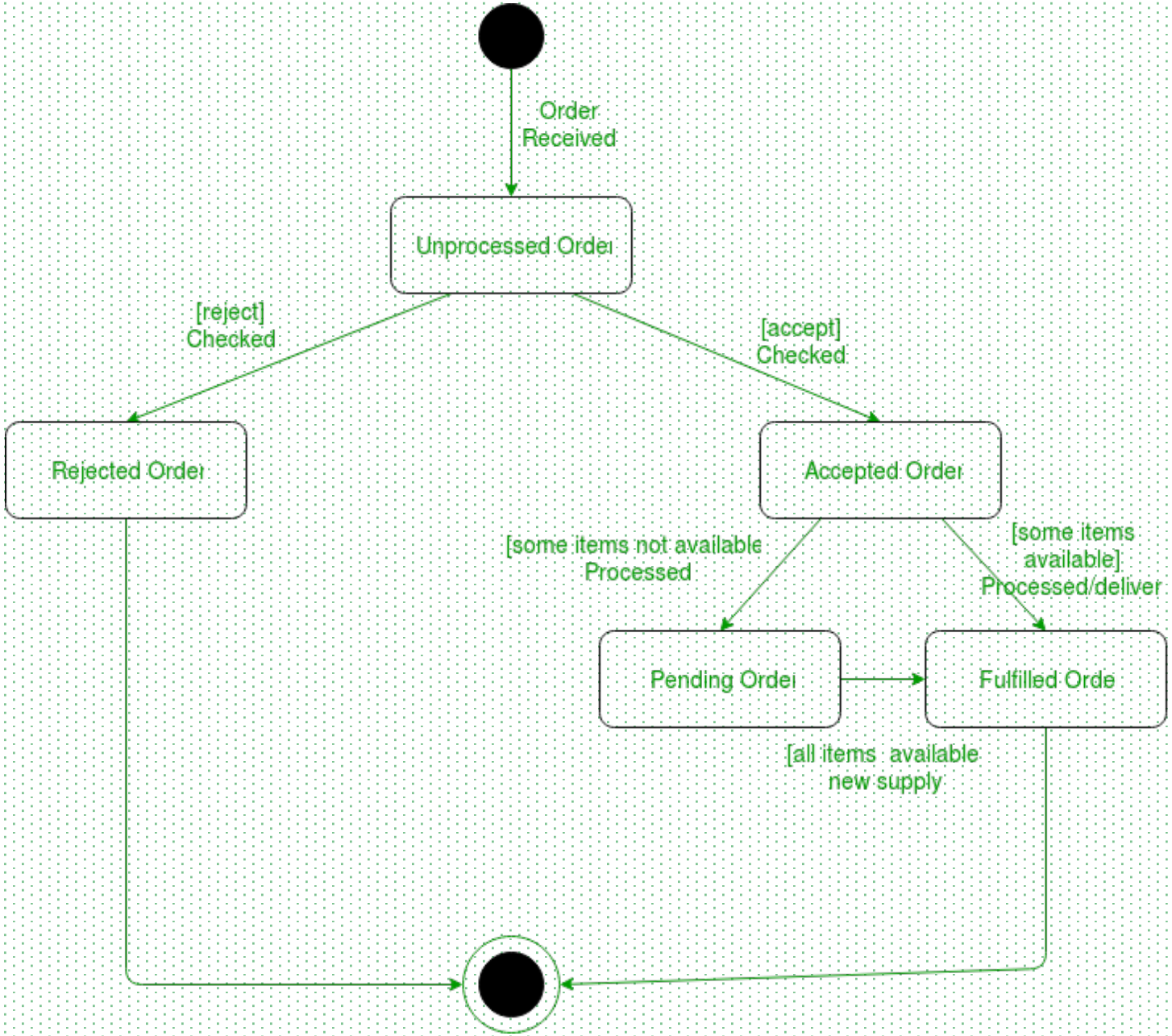


Birleştirme durumlarından gelen okları ve ortak hedef durumuna giden okları içeren bir Join gösterimini temsil etmek için yuvarlak, içi dolu bir dikdörtgen çubuk kullanırız. Bir olay veya olayın meydana gelmesinde iki veya daha fazla durum aynı anda bire yakınsadığında birleştirme notasyonunu kullanırız.



Kendinden geçişi temsil etmek için,bir olayın meydana gelmesi üzerine nesnenin durumunun değişmediği senaryolar olabilir. Bu gibi durumları temsil etmek için kendi kendine geçişleri kullanırız.

Durum diyagramı UML örneği



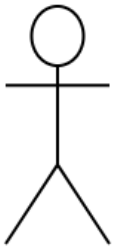
4. Sıra diyagramı (Sequence Diagram)

- Bir sıra diyagramı, nesneler arasındaki etkileşimi basitçe sıralı bir düzende, yani bu etkileşimlerin gerçekleştiği sırayı gösterir.
- Sıra diyagramları, bir sistemdeki nesnelerin nasıl ve hangi sırayla çalıştığını açıklar. Bu diyagramlar, iş adamları ve yazılım geliştiriciler tarafından yeni ve mevcut sistemlerin gereksinimlerini belgelemek ve anlamak için yaygın olarak kullanılmaktadır.

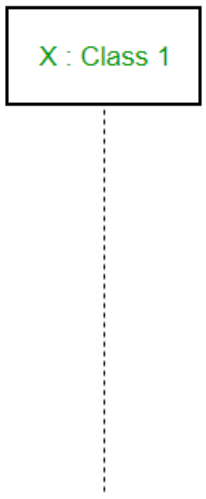
Sequence diagramlarının kullanım amaçları

- Sofistike bir işlemin veya prosedürün arkasındaki mantığı modellemek ve görselleştirmek için kullanılır.
- Ayrıca UML kullanım durumu diyagramlarının ayrıntılarını göstermek için kullanılırlar.
- Mevcut veya gelecekteki sistemlerin ayrıntılı işlevlerini anlamak için kullanılır.
- Mesajların ve görevlerin bir sistemdeki nesneler veya bileşenler arasında nasıl hareket ettiğini görselleştirmek için kullanılır.

Sıra (Sequence) diagram gösterimleri

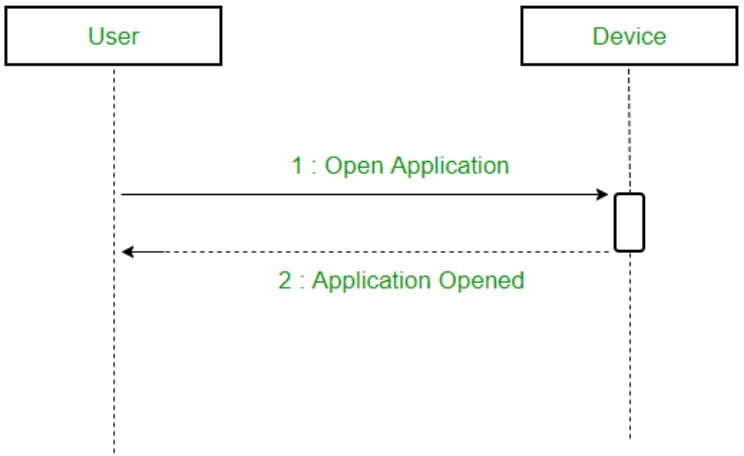


Aktör – Sistem ve nesneleri ile etkileşime girdiği bir rol türünü temsil eder. Burada bir aktörün her zaman UML diyagramını kullanarak modellemeyi hedeflediğimiz sistemin kapsamı dışında olduğuna dikkat etmek önemlidir.



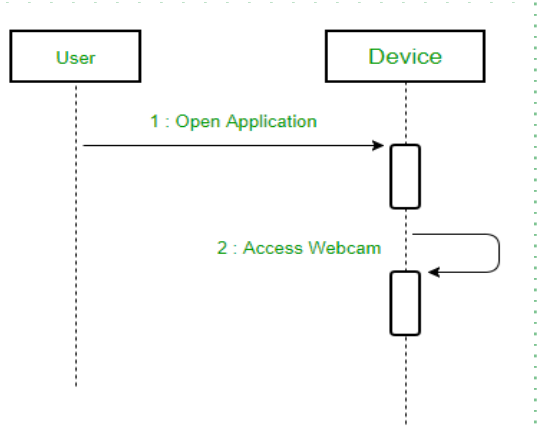
Yaşam çizgileri – Yaşam çizgisi, bir dizi diyagramında bireysel bir katılımcıyı gösteren adlandırılmış bir öğedir. Yani temel olarak bir dizi diyagramındaki her örnek bir yaşam çizgisi ile temsil edilir. Yaşam çizgisi elemanları bir dizi diyagramında en üstte yer alır.

Eşzamanlı (senkron) mesajlar – Eşzamanlı bir mesaj, etkileşimin ilerleyebilmesi için bir yanıt bekler. Gönderici, alıcı mesajın işlenmesini tamamlayana kadar bekler. Senkronize bir mesajı temsil etmek için düz bir ok başı kullanıyoruz.

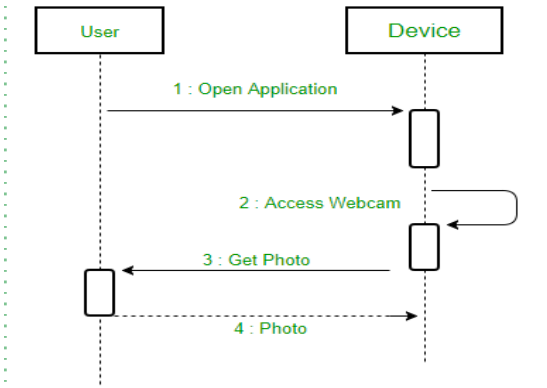


Asenkron mesajlar – Asenkron bir mesaj, alıcıdan bir cevap beklemez. Etkileşim, alıcının önceki mesajı işleyip işlemediğinden bağımsız olarak ilerler.

Self Mesaj – Nesnenin kendisine bir mesaj göndermesi gereken belirli senaryolar ortaya çıkabilir. Bu tür mesajlara Kendi Mesajları denir ve U şeklinde bir okla gösterilir.

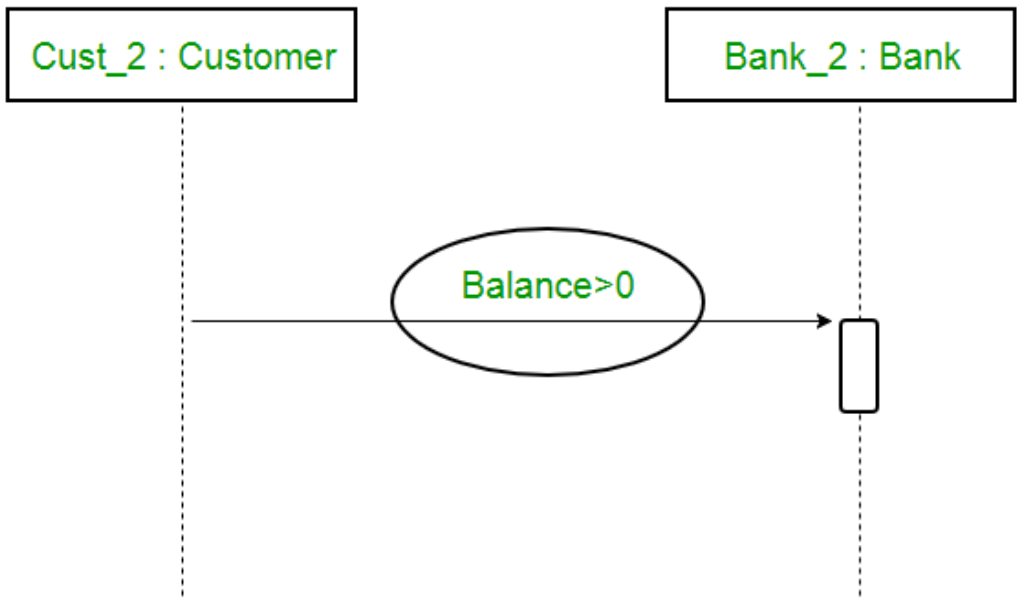


Reply Mesajı – Cevap mesajları, alıcıdan gönderene gönderilen mesajı göstermek için kullanılır. Noktalı bir çizgi ile açık bir ok ucu kullanan bir geri dönüş/yanıt mesajını temsil ediyoruz. Etkileşim, yalnızca alıcı tarafından bir yanıt mesajı gönderildiğinde ilerler.

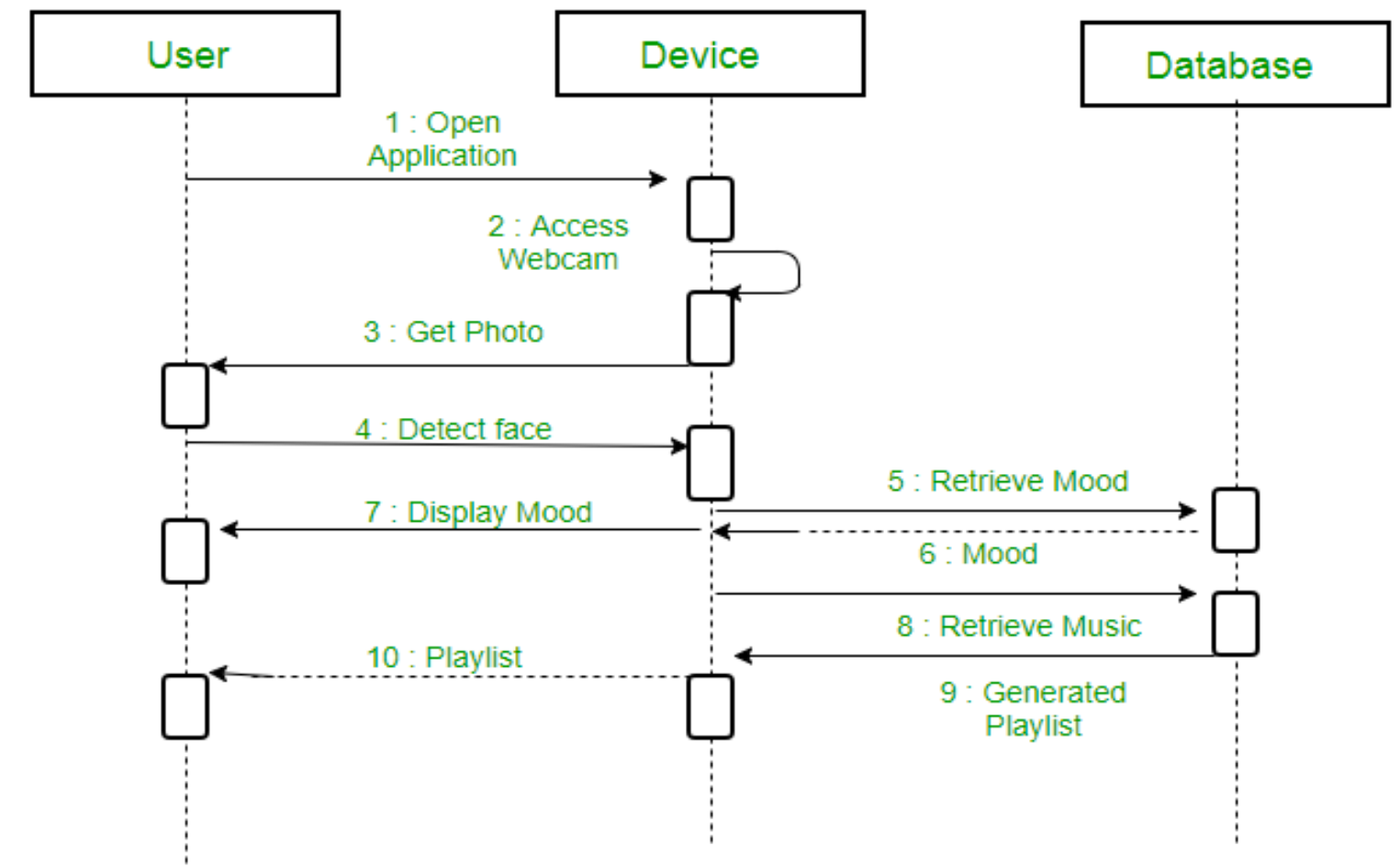


Guard – Koşulları modellemek için UML'de korumalar kullanırız. Karşılanan bir koşul bahanesiyle mesaj akışını kısıtlamamız gerektiğinde kullanılırlar. Korumalar, yazılım geliştiricilerin bir sisteme veya belirli bir sürece bağlı kısıtlamaları bilmelerine izin vermede önemli bir rol oynar.

Örneğin: Nakit çekebilmek için sıfırdan büyük bir bakiyeye sahip olmak, aşağıda gösterildiği gibi yerine getirilmesi gereken bir koşuldur.



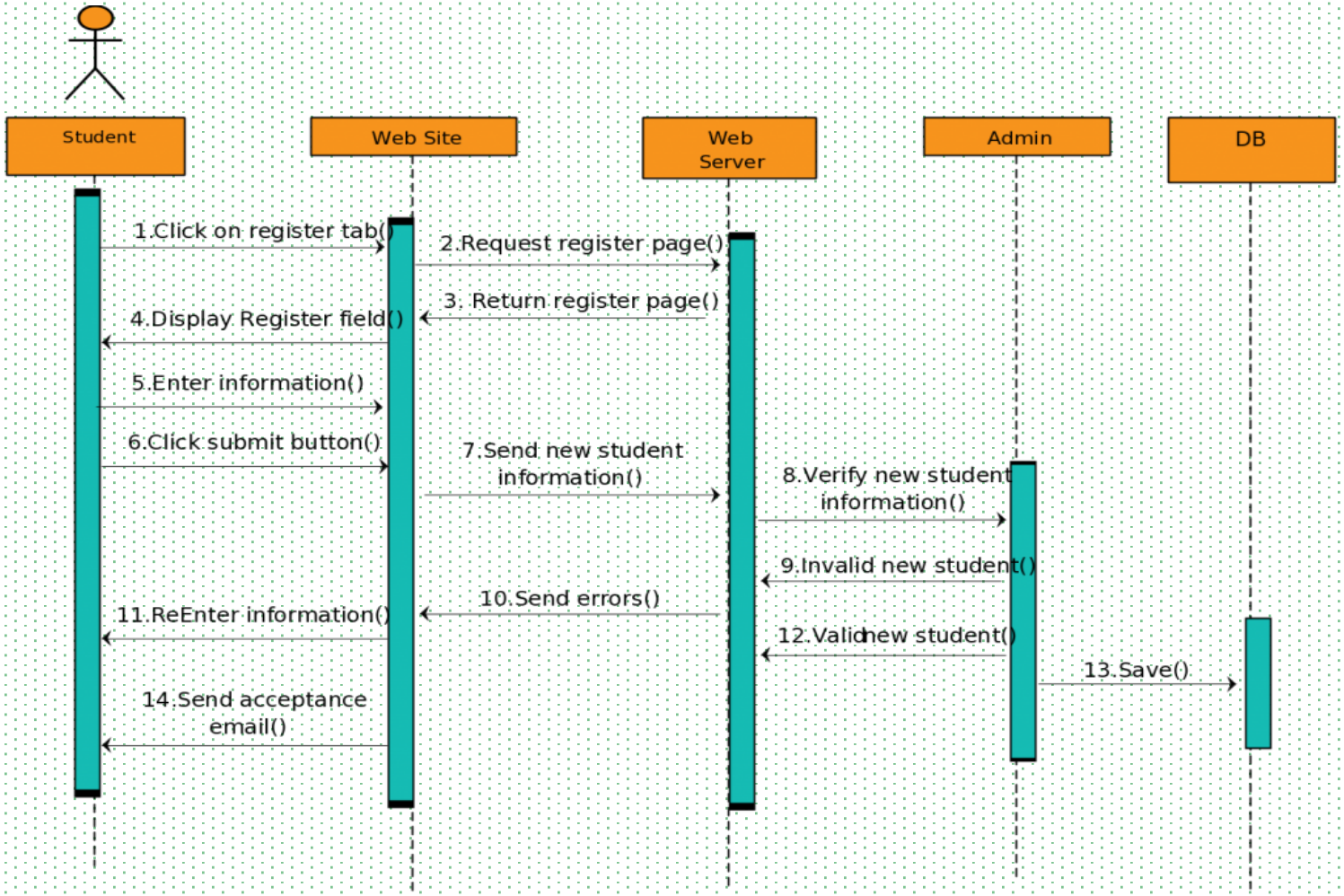
Sequence diagram örneği
Duygu tabanlı bir müzik çalar



Yukarıdaki sıra şeması, duygu tabanlı bir müzik çaların sıra şemasını göstermektedir:

1. Öncelikle uygulama kullanıcı tarafından açılır.
2. Cihaz daha sonra web kamerasına erişim sağlar.
3. Web kamerası, kullanıcının görüntüsünü yakalar.
4. Cihaz, yüzü algılamak ve ruh halini tahmin etmek için algoritmalar kullanır.
5. Daha sonra olası ruh halleri sözlüğü için veri tabanı talep eder.
6. Ruh hali veritabanından alınır.
7. Ruh hali kullanıcıya gösterilir.
8. Müzik veritabanından isteniyor.
9. Çalma listesi oluşturulur ve son olarak kullanıcıya gösterilir.

Sequence diagramı örnek2



5. İletişim diyagramı (Communication Diagram)

- Communication diagramı (UML 1.x'te Collaboration şeması olarak bilinir. Nesneler arasında değiş tokuş edilen sıralı mesajları göstermek için kullanılır. Bir communication diyagramı öncelikle nesnelere ve onların ilişkilerine odaklanır. Sıra diyagramlarını kullanarak benzer bilgileri gösterebiliriz, ancak iletişim diyagramları nesneleri ve bağlantıları serbest bir biçimde temsil eder.

Sequence ve Communication diagramlarının farkları

Sequence Diagramı	**Communication yada Collaboration**
Sıra diyagramı, belirli bir işlevi gerçekleştirmek için kullanılan bir sistemdeki çağrı sırasını görselleştirmek için kullanılan UML'yi temsil eder.	İşbirliği diyagramı ayrıca nesnelerin organizasyonunu ve etkileşimlerini görselleştirmek için kullanılan UML temsilinin altına gelir.
Sıra diyagramı, bir nesneden diğerine akan mesajların sırasını temsil etmek için kullanılır.	İşbirliği diyagramı, sistemin yapısal organizasyonunu ve gönderilen ve alınan mesajları temsil etmek için kullanılır.
Dizi diyagramı, zaman dizisi birinci planda olduğunda kullanılır.	İşbirliği diyagramı, nesne organizasyonu ana odak olduğunda kullanılır.
Sıra diyagramları, analiz faaliyetleri için daha uygundur.	İşbirliği diyagramları, daha az sayıda nesnenin daha basit etkileşimlerini göstermek için daha uygundur.

5. Zamanlama diyagramı(Timing Diagram)

- Zamanlama Diyagramı, nesnelerin bir zaman çerçevesi içindeki davranışını göstermek için kullanılan özel bir sıra diyagramıdır. Bunları, nesnelerin durumlarındaki ve davranışlarındaki değişiklikleri yöneten zaman ve süre kısıtlamalarını göstermek için kullanırız.

Kaynaklar

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş”, TBİL-211, Dr. Ali Arifoğlu.

”Yazılım Mühendisliği” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)

https://www.mustafayemural.com/

https://tugrulbayrak.medium.com/

Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

Duygu Çelik, Doğu Akdeniz Üniversitesi

Fırat Yeşilyürdü, Model güdümlü mimari kullanılarak bir konum sunucusu yazılımının geliştirilmesi.

Sorularınız