ARDIŞIL DİYAGRAM – YAPI DİYAGRAMI

Sistem Analizi ve Tasarımı Dersi

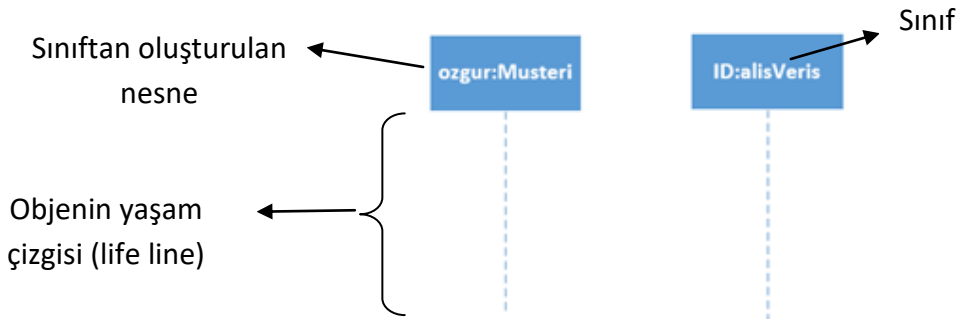
İçindekiler

Ardışıl Diyagram Nedir ve Neden Kullanılır

Ardışıl diyagram (Sequence diagram) nesnelerin birbirleriyle etkileşimini ve etkileşimin sırasını gösteren bir UML diyagramıdır. Literatürde olay diyagramı diye de geçmektedir (event diagram). Ardışıl diyagramlar nesneler (sınıflar/kavramlar) arasındaki mesaj, aksiyon ve olay akışını detaylı olarak kullanıcıya aktarmakta kullanılır. Nesneler arasındaki etkileşimler ardışıl diyagramda ardışıl zamanlarla gösterilerek ifade edilir. Sınıf diyagramından farklı olarak ardışıl diyagramlar nesneler arasındaki etkileşimi zaman sıralamasına göre kullanıcıya sunar. Birbirleriyle etkileşim halindeki bilgi/olay akışının kullanıcıya sunulmasıyla nesneler arasındaki etkileşim (mesajlaşma) detaylıca sunulur. Mesajlar akış diyagramında oluşturuldukları sıraya göre yukarıdan aşağıya doğru çizilir. Ardışıl diyagramın problemi karmaşık sistemlerde nesneler arası mesajlaşma yoğunluğu fazla olacağı için tüm bu mesajlaşmanın görselleştirilmesi zaman alan ve dokümantasyon olarak çok yer kaplayan bir iştir.

Ardışıl Diyagram Elemanları

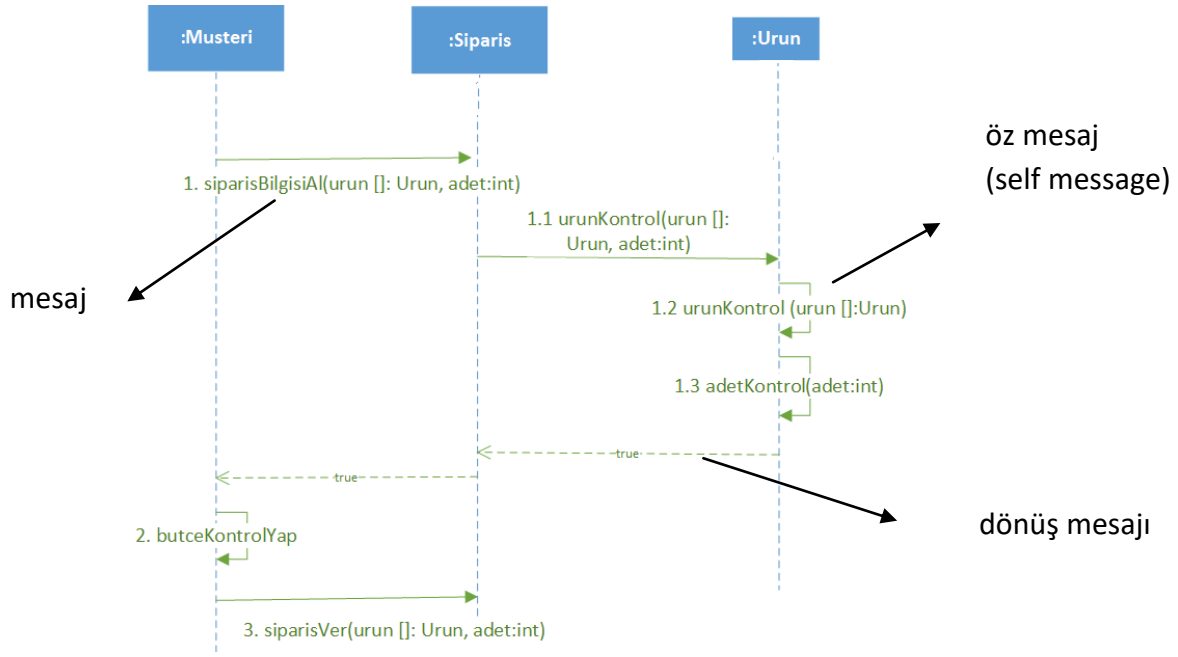
- Katılan (Paydaş veya İştirakçi): Bir nesne veya varlıktır. Şekil 1'de örnek verilmiştir.



Şekil 1. Nesne ve yaşam çizgisi

- Eksenler: Dikey eksen hangi paydaşların etkin olduğunu gösterirken, yatay eksen zamanı göstermektedir.

- Mesaj: Nesneler arasındaki iletişimi gösterir. Şekil 2'de farklı mesaj tipleri verilmiştir.

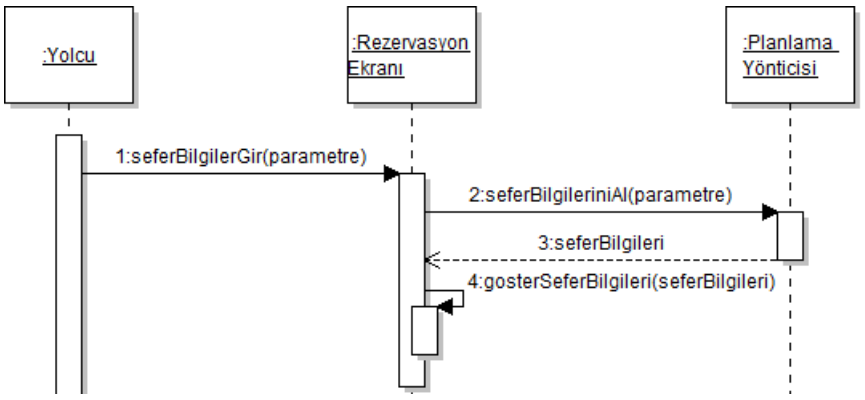


Şekil 2. Mesaj, öz mesaj, dönüş mesajı ve eksenler

Örnek Senaryo :

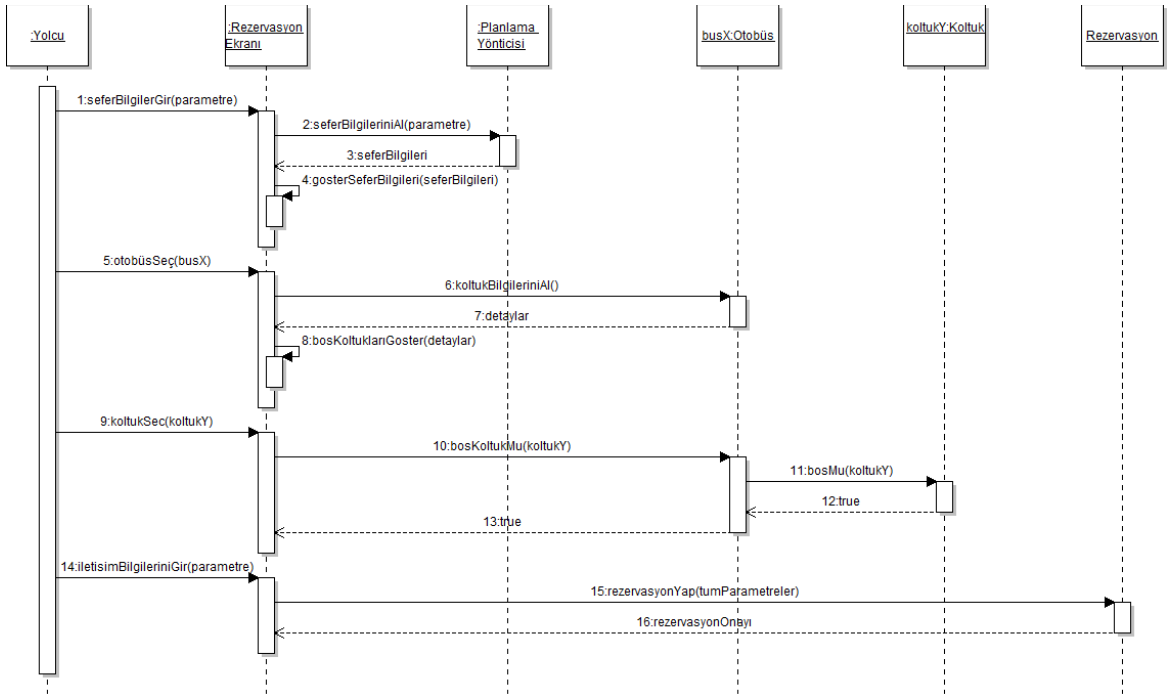
Bir yolcunun yolculuk etmek istediği seferi, istediği seferle ilgili otobüsü ve ilgili otobüsteki koltuğu seçmesini sağlayan ve ilgili bilgilerin rezervasyon bilgisi olarak kaydedildiği bir sistem tasarlanmak istenmektedir. Senaryonun ana akışı gösteren adımlar aşağıda belirtilmiştir.

1. Yolcu seyahat yapmak isteyeceği seyahat bilgilerini girer ve bu bilgilere ait otobüs seferleri sistem tarafından gösterilir.
2. Yolcu bir otobüs seçer.
3. Yolcu seçilen otobüste bir koltuk seçer ve sistem koltuğun uygunluğunu kontrol eder.
4. Yolcu seçimlerini yaptıktan sonra rezervasyon bilgilerini girer ve sistem kaydı gerçeklenir.



Şekil 3. Ardışıl diyagram adımı

Şekil 3'e göre nesne (sınıf) olan yolcu sisteme sefer bilgileri ara yüz olan ekrandan girerek objeler arası etkileşimi başlatmaktadır. Senaryoda aynı saatte aynı kalkış ve istikamete birden fazla otobüs olacağı öngörülerek senaryonun ana akışında varlık olarak bulunmamasına rağmen "PlanlamaYöneticisi" adında yeni bir sınıf tanımlanmıştır. Şekil 3'te sefer bilgileri rezervasyon ekranına yansıtılır ve bu işlem için "RezervasyonEkranı" sınıfı öz metodunu kullanır.



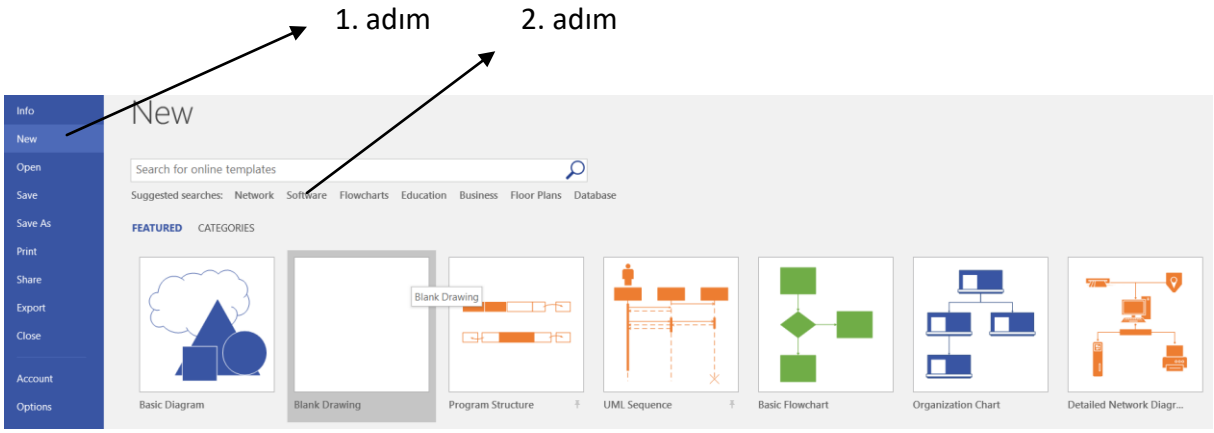
Şekil 4. Senaryonun bütün ardışıl diyagramı

Kullanıcı sefer bilgisi gördükten sonra seçtiği ve parametre olarak gönderdiği (*busX*) koltuk bilgilerini alır ve koltuk seçiminde bulunur. Bu işlemleri tamamladıktan sonra iletişim bilgileri girerek rezervasyon işlemini tamamlar. Şekil 4'te tüm akış sırası mesajlar ile detaylıca gösterilmiştir.

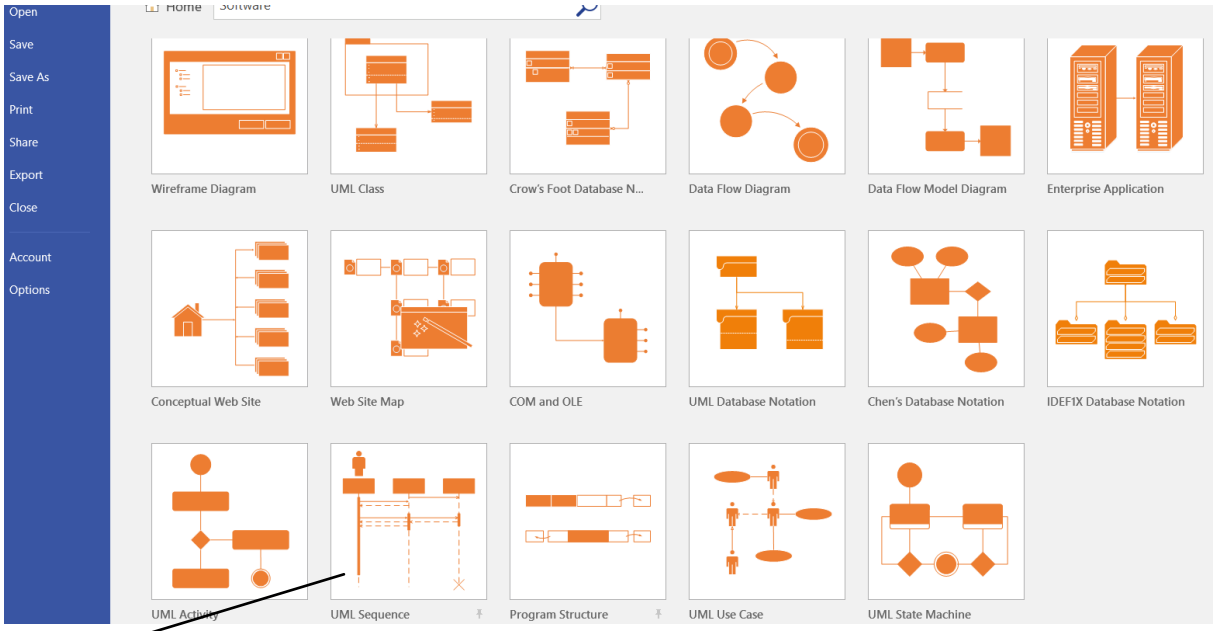
MS Visio ile Ardışıl Diyagram Çizimi

MS Visio 2016 ile Ardışıl Diyagram çizimi için:

New ---> Software ----> UML Sequence---> Create

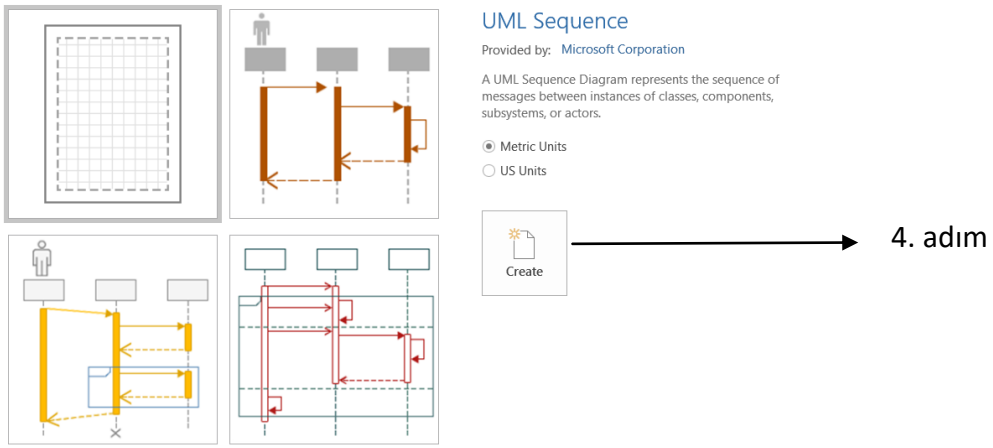


Şekil 5. MS Visio 2016 ile yeni ardışıl diyagram oluşturma- Adım 1



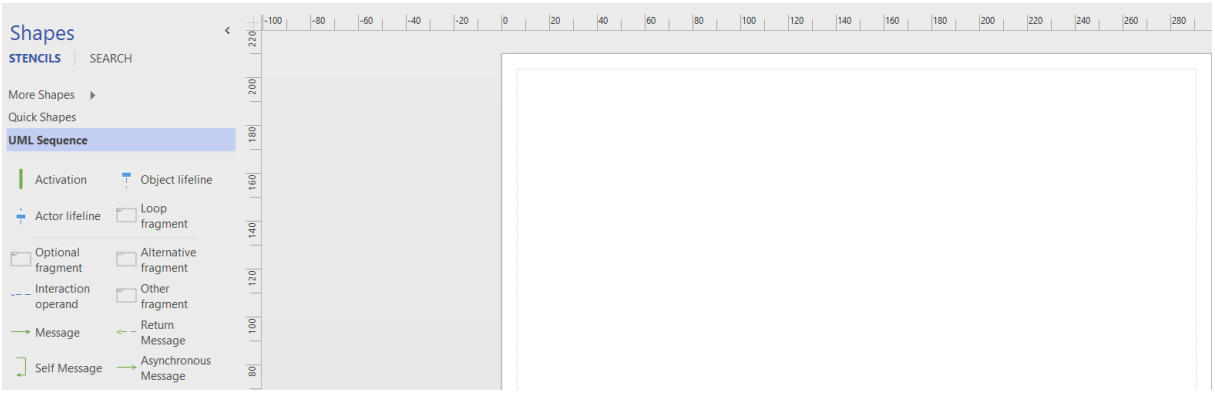
Şekil 6. MS Visio 2016 ile yeni ardışıl diyagram oluşturma - Adım 2

3. adım



Şekil 7. MS Visio 2016 ile yeni ardışıl diyagram oluşturma - Adım 3

Şekil 5-7'de yer alan alanları sırasıyla kullanıcının tıklamasıyla yeni bir Ardışıl Diyagram dosyası oluşturulmuş olacaktır. Sırasıyla Şekil 5'te *New --->Software* ile seçerek yazılım alanındaki diyagramları görmekte olan kullanıcı Şekil 6'daki gibi *UML Sequence* ile ardışıl diyagramı seçer. Şekil 7'deki son işlem olan *Create* adımıyla yeni bir ardışıl diyagram oluşturur.

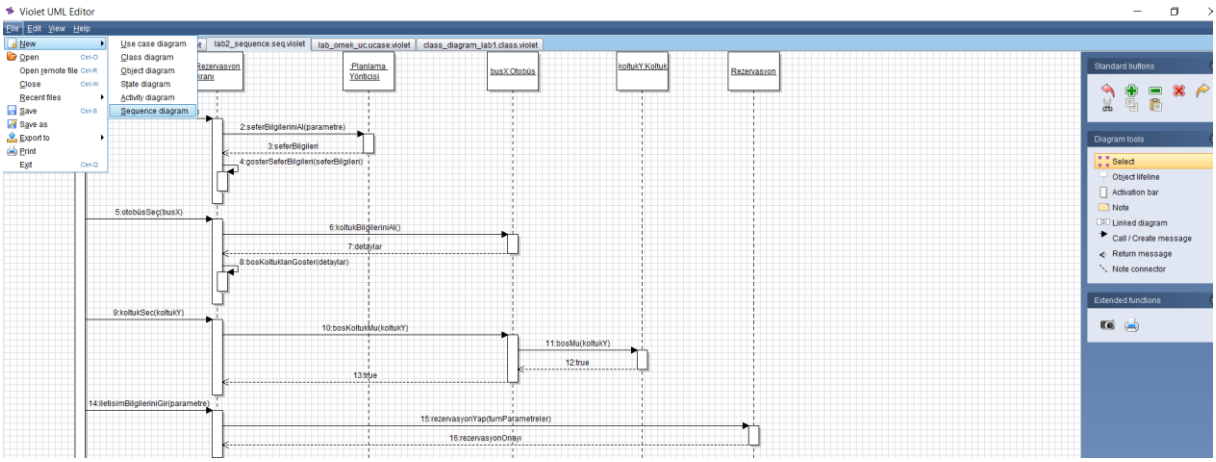


Şekil 8. MS Visio 2016 ile ardışıl diyagram oluşturma

Şekil 8'de "Object Lifeline" sürükle bırakla aktörler (paydaş), "Message" kontrolü sürükle bırakla mesajlar, "Self Message" sürükle bırakla öz mesajlar, "Return Message" kullanarak ise dönüş mesajları nesneler için tanımlanabilmektedir.

Violet UML ile Ardışıl Diyagram Çizimi

Violet UML programı açıldıktan sonra kullanıcının yeni bir ardışıl diyagram oluşturmak için yapması gerekenler Şekil 9'da verilmiştir. Kullanıcı New---> Sequence diagram menüsünü kullanarak yeni bir ardışıl diyagram dosyası oluşturabilmektedir.



Şekil 9. Violet UML ile ardışıl diyagram oluşturma

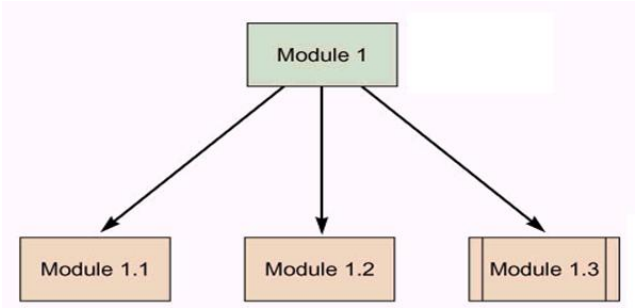
Violet UML kullanarak Ardışıl Diyagram çizilirken mesaj oluşturmak için "Call/Create Message", obje ve yaşam çizgisi oluşturmak için "Object Lifeline" ve dönüş mesajı için "Return Message" kontrolleri kullanılmaktadır.

Yapı Diyagram Nedir ve Neden Kullanılır

Yapı diyagramı sistemdeki (yazılımdaki) yönetilebilen seviyedeki her bir parçanın (modülün) fonksiyonunu görüntülemek için kullanılan diyagramdır. Modüllerin her biri spesifik bir görevi (fonksiyonu) gerçekleştirmektedir. Yapısal programlamada (structured programming) kullanılan bu diyagramlar program modülleri bir ağaç yapısı şeklinde gösterilir. Her bir modül bir dikdörtgen şeklinde gösterilmekle birlikte modülün adı dikdörtgen içerisinde yazmaktadır. Ağaç yapısı modüller arası ilişkileri ve akışı göstermektedir.

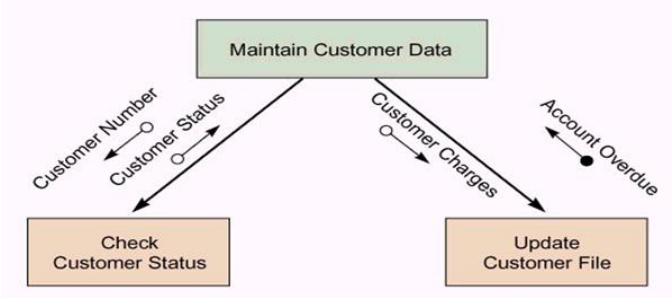
Yapı Diyagram Elemanları

- Modül (program veya alt program): Şekil 10'daki "Module 1" isimli modül alt modüllere ulaştığı için kontrol modülüdür (control module). "Module 1.3" isimli modül birden fazla yerden ve farklı kontrol modülleriyle ilişkili olabileceği için bu tarz modüller kütüphane modülü (library module) olarak adlandırılmışlardır.



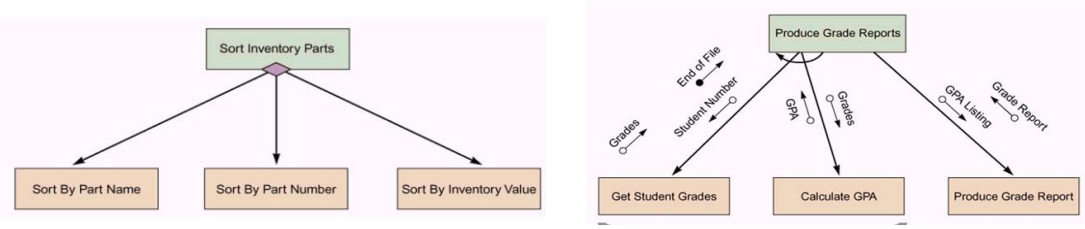
Şekil 10. Yapı diyagramında modül kontrolü

- Veri (data): Şekil 11'deki içi boş olan oklar akışları gösterir. Okun yönü aynı zamanda akışın da yönüdür. İçi dolu olan oklar ise kontrol akışıdır ve işlemin tamamlandığını ve/veya işlem onayını gösterir.



Şekil 11. Yapı diyagramında veri kontrolü

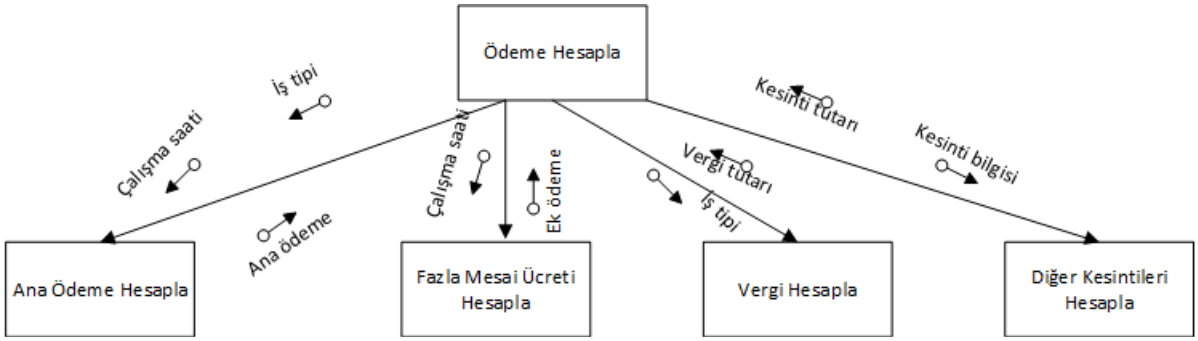
- Döngü ve Durum Kontrolü: Şekil 12'deki gibi yapı diyagramında durum kontrolü elmas şeklinde, döngü ise kavisli ok şeklinde gösterilmektedir.



Şekil 12. Yapı diyagramında döngü ve durum kontrolü

Şekil 12'de envanter parçalarının sıralamasındaki farklı seçenekler durum kontrolüyle gösterilmiştir. Durum kontrolü kontrol modülünün alt modülü aktive ederken farklı seçenekleri olduğunu göstermektedir. Şekil 8 aynı zamanda döngü modülünün kullanımını kullanıcıya açıklamaktadır. Öğrencinin numaraları alındıktan sonra öğrencinin tüm notları elde ediliyor ve her bir öğrenci için genel not ortalaması hesaplanmaktadır. Bu tüm öğrenciler için gerçekleştirildiği için döngü kontrolü kullanılmıştır.

Örnek Senaryo 1:



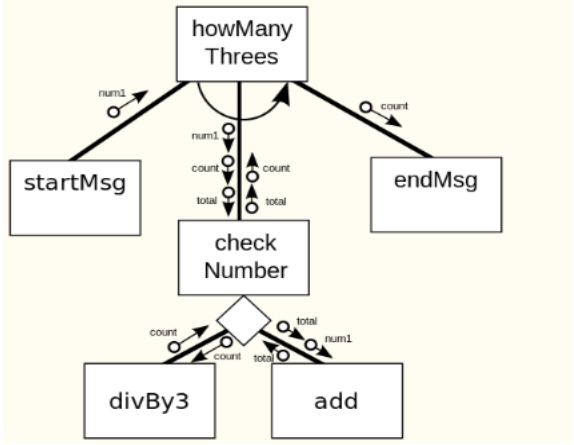
Şekil 13. Yapı diyagramı örnek senaryo

Şekil 13'te "Ödeme Hesapla" kontrol modülünün altında 4 farklı modül bulunmaktadır. "Ödeme Hesapla" ana modülü alt modülleri kullanarak fonksiyonellik kazanmaktadır. Bu alt modüllerin aldıkları verileri ve ürettikleri veriler şekilde gösterilmiştir. Yapısal programlamada alınan veriler fonksiyonlardaki parametrelere, ürettikleri veriler ise dönüş değerine (return value) denk gelmektedir. Buna göre "Ana Ödeme Hesapla" alt modülüne çalışma saati ve iş tipi parametrelerini alarak ana ödeme değerini hesaplamakta ve bu değeri döndürmektedir. 4 alt modülü kullanarak kontrol modülü fonksiyonu yerine getirilmiş olur. Senaryoda yer alan 4 modül birbirinden bağımsız olarak belirtilmiştir. Yapı diyagramlarında iş akışı aynı seviyedeki alt programlar soldan sağa doğru olmakla birlikte alt programlar birbirinden bağımsız olabilmektedir.

Örnek Senaryo 2: Yapısal programlamada kaynak kod kullanarak yapısal diyagramlar elde edilebileceği gibi, yapısal diyagramlardan da taslak kaynak kodlar elde edilebilir. Şekil 14'te bu duruma örnek bir senaryo verilmiştir.

```
sub howManyThrees()
  dim num1, count, total as integer
  num1 = startMsg()
  count = 0
  total = 0
  while num1 > 0 do
     checkNumber(count, total, num1)
     num1 = num1 - 1
  end while
  endMsg(count)
end sub

sub checkNumber(byRef c, byRef t, byVal n)
  If n MOD 3 = 0 Then
     c = divBy3(c)
  Else
     t = add(n, t)
  EndIf
end sub

function divBy3(x)
  return x + 1
end function

function add(n, t)
  return n + t
end function

function startMsg()
  console.writeline("program started, enter your number")
  return console.readline()
end function

sub endMsg(n)
  console.writeline("number of threes : " & n)
end sub
```

Answer :



Şekil 14. Yapı diyagramı- kaynak kod

Şekil 14'teki yapı diyagramı kod ilişkisine göre "howManyThrees" fonksiyonu "startMsg" ile kullanıcıdan alınan değerin içerisinde kaç tane üçün geçtiğini gösteren bir fonksiyondur. Sonunda elde edilen değer "endMsg" fonksiyonu ile yazdırılmaktadır. Buna göre " howManyThrees" fonksiyonundaki alt fonksiyon olan "checkNumber" fonksiyonu ilk değer olarak ilk iki parametreyi sıfır, son parametreyi de içerisinde üç bulacağı sayı olarak alır. Üçüncü parametre değerini bir döngü içerisinde sıfıra eşit olana kadar azaltılır. Bu azaltma işlemlerinde önce kullanıcıdan alınan sayı üçe tam olarak bölünüyorsa ilk parametre değeri (ilk değeri sıfırdı) bir arttırılır ve dönüş değeri olarak belirlenir. Eğer azaltılan sayı üçe bölünmüyorsa ikinci ve üçüncü parametre toplanarak dönüş değeri belirlenir. Döngü sonunda kullanıcıdan elde edilen ilk parametre değeri yani kullanıcıdan alınan sayının

içerisinde kaç defa üçün geçtiğini gösteren değer "endMsg" ile yazdırılarak kontrol fonksiyonu işlevini yerine getirmiş olur.

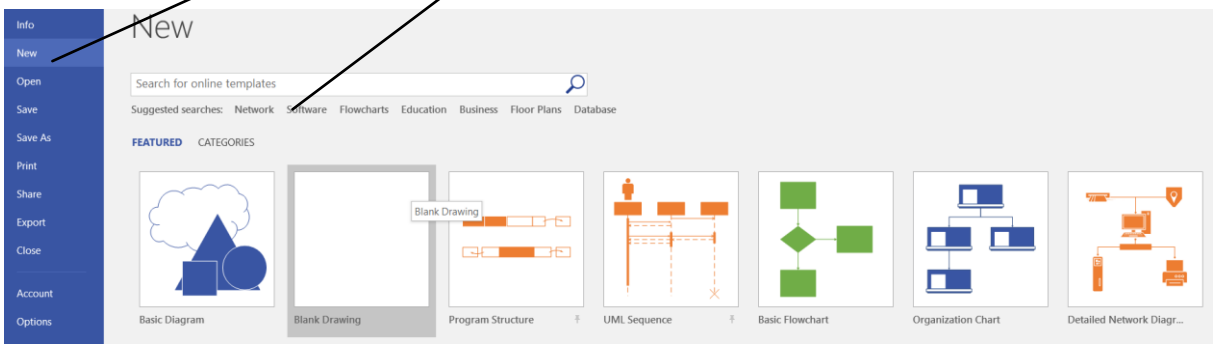
MS Visio ile Yapı Diyagram Çizimi

MS Visio 2016 ile Ardışıl Diyagram çizimi için:

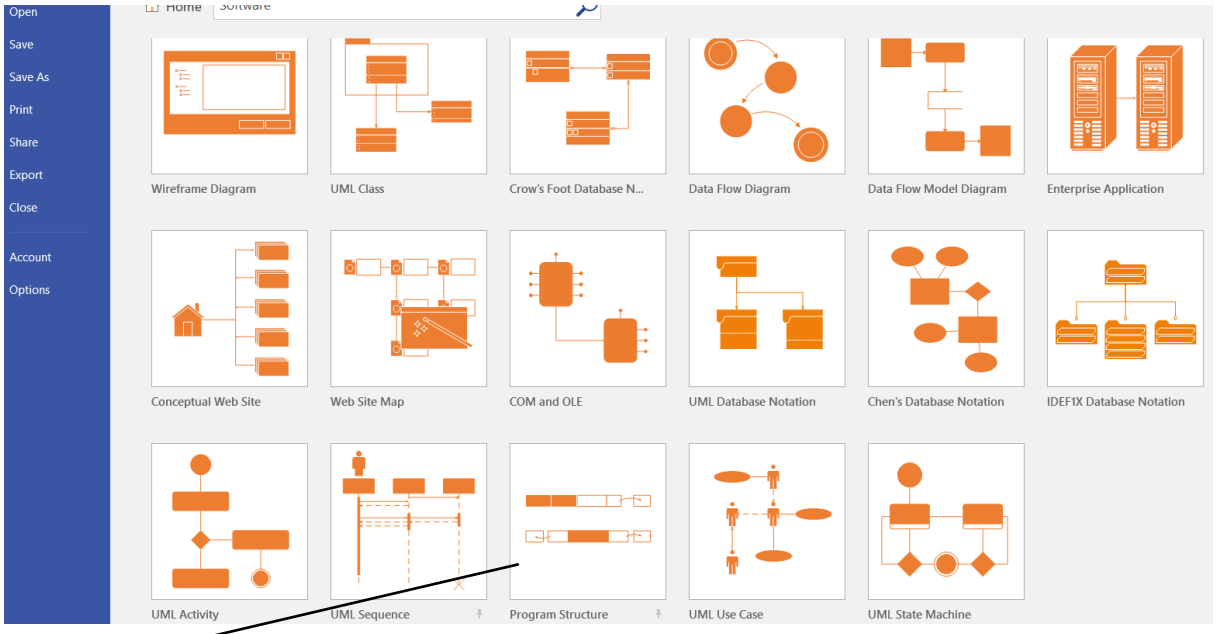
New ---> Software ----> Program Structure---> Create

1. adım

2. adım

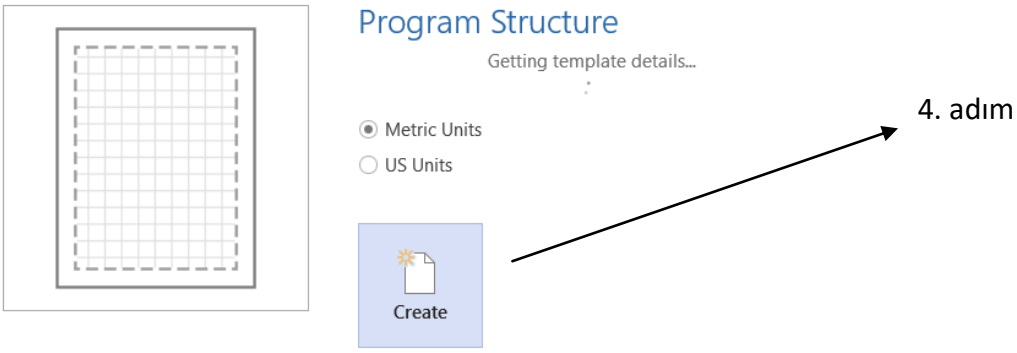


Şekil 15. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 1



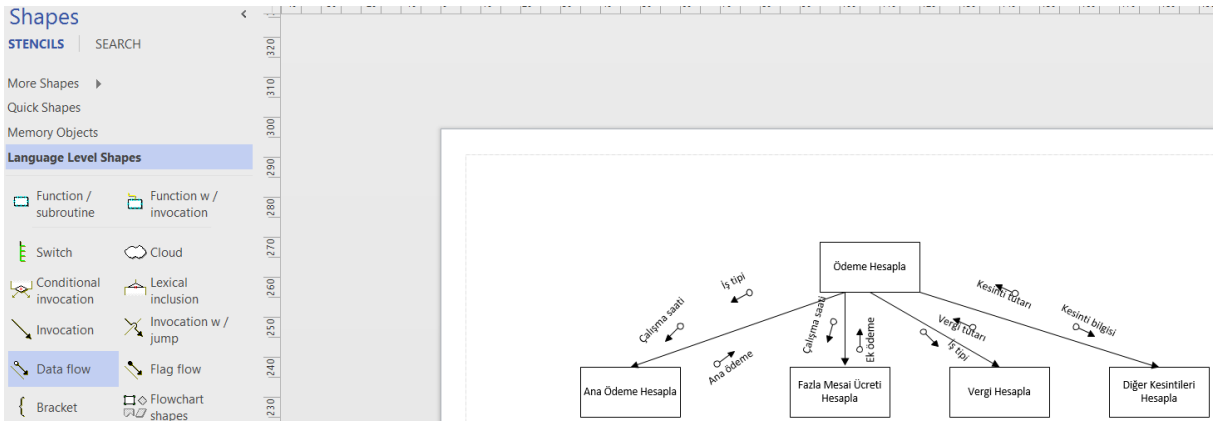
Şekil 16. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 2

3. adım



Şekil 17. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 3

Şekil 15-17'de yer alan alanları sırasıyla kullanıcının tıklamasıyla yeni bir Yapı Diyagram dosyası oluşturulmuş olacaktır. Sırasıyla Şekil 15'te *New --->Software* ile seçerek yazılım alanındaki diyagramları görmekte olan kullanıcı Şekil 16'daki gibi *Program Structure* ile yapı diyagramını seçer. Şekil 17'deki son işlem olan *Create* adımıyla yeni bir yapı diyagramı oluşturur.



Şekil 18. MS Visio 2016'daki yapı diyagramı kontrolleri

Şekil 18' de gösterilmiş kontrollerin detayları aşağıda kısaca özetlenmiştir. Buna göre :

- Function/subroutine: Kontrol modülü
- Function w/ invocation: Alt modül
- Invocation: Bağlantılar
- Data flow: Veri akışı
- Flag flow: Kontrol akışı
- Flowchart Shapes: Döngü ve diğer kontroller
- Conditional Invocation: Durum kontrolü