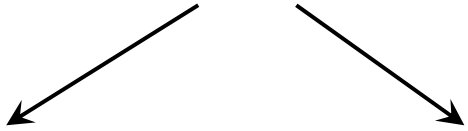
SIMAN

Simülasyon (Benzetim) Modeli Türleri:



İconic (Simulators)	**Symbolic**
Genellikle eğitim amaçlı kullanılırlar. Örneğin; Uçuş Simülatörleri (nasıl uçulduğunu ve acil bir durumda ne yapılacağı öğretilir	Bu modellerle sistem, matematiksel açıklamalar seti ve mantıksal ilişkilerle ifade edilir. Bu modeller genelde bilgisayarda koşturulur.

SIMAN Benzetim Dili:

SİMulation ANalysis

Siman, kesikli, sürekli veya kesikli-sürekli sistemlerin modellenmesi için geliştirilmiş genel amaçlı bir benzetim dilidir.
Kesikli-değişen sistemler proses etkileşimi ile veya olay-programlama ile modellenir.
Sürekli sistemler algebra, fark veya diferansiyel eşitliklerle modellenir.
Bunların kombinasyonu ile kesikli-sürekli modellerin benzetimi yapılır.

SIMAN'da benzetim problemi iki segment halinde modellenir.

1) “model” bileşeni
2) “experiment” bileşeni

Bu genel çerçeve zeigler (1976) tarafından geliştirilen sistemler hakkındaki genel teorik kavramlara dayanmaktadır.
Bir model; sistemin makinalar, işçiler, depolama yerleri, taşıyıcılar, parça akışı, bilgiler gibi fiziksel elemanlarını ve onların mantıksal ilişkilerini açıklar.
Experiment; modelin koşturulacağı deneysel koşulları belirler. Başlangıç koşulları kaynak durumunu toplanan istatistik türünü koşum uzunluğunu da kapsar.
“Model” ve “experiment” tanımlandığı zaman SIMAN ile aralarındaki bağ oluşturulur. Ve execute edilir. Benzetim çalıştırıldıktan sonra SIMAN experiment içinde belirlenen sonucu (cevabı) saklar.

SIMAN’ın output çözümleyicisi çizimler, tablolar Barchartlar, histogramlar, güven aralıklarının oluşturulmasında kullanılır.

BENZETİM DİLİ KULLANMANIN AVANTAJLARI

1) Programlama işlemini azaltır, kolaylaştırır.
2) Hata bulma daha kolaydır.
3) Model değiştirme veya güncelleme daha kolaydır.

GENEL AMAÇLI DİL KULLANMANIN AVANTAJI

1) Genel amaçlı diller hemen hemen her bilgisayarda kullanılabilir.
2) Çoğu insan bir genel amaçlı dil bilir fakat benzetim dili bilmeyebilir.
3) Genel amaçlı diller esnek programlamaya imkan verirler. (benzetim diline göre)

4) Etkin olarak genel amaçlı bir dille yazılmış program daha hızlıdır.

Tanımlar:

Entity: Sistem boyunca akan (hareket eden) birim elemanlardır. Bir kişi, bir obje veya herhangi bir şey bir "entity"dir ve sistemin durumunu değiştirir. Verilen bir sistem içinde birden fazla entity kullanılabilir.

Attributes: "Entitiy"lerin karakteristik özellikleridir. Örneğin bir fabrikada entity=iş parçası olsun. Attributes ise parça nosu, teslim tarihi ve iş parçasının önceliği olabilir.

SIMAN'da "entity" dinamiktir. Entity" nin modele girişi ve çıkışı, objenin sisteme varışı ve çıkışına karşılıkdır.
Model içindeki entity sayısı her an değişmektedir. (Yeni bir entiti her an sisteme girebilir veya mevcut bir entiti çıkabilir.)

"Process" terimi SIMAN'da işlemler veya aktivitelerin bir sıralaması demektir. Entitiler bu sıralama içinde hareket ederler. Prosessi aktif hale getirmek için bir entity gerekir.

BLOCK DİYAGRAMLARI

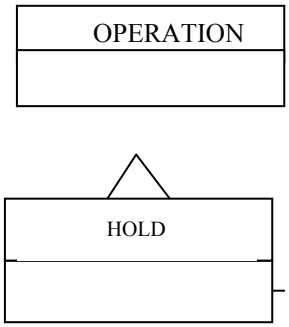
SIMAN'da process "blok" diyagramları kullanılarak modellenir. Blok diyagramı doğrusal ve yukarıdan-aşağıya doğru (top-down flow) akışlıdır.

Blokların bir sıralaması olarak oluşturulan "blok diyagramı" içinde kullanılan blok şekilleri genel fonksiyonlarını belirtir. Sıralama ↓ akış oklarıyla belirlenir. Bu oklar, model içinde bir entity"nin bir bloktan diğer bloğa hareket yönünü gösterir.

TEMEL BLOK TÜRLERİ

SIMAN'da 40'ın üzerinde farklı proses modelleme fonksiyonlarını tanımlayan blok vardır. Bunlar azaltılarak temelde 10 blok türüne indirilmiştir.

CREATE, CLOSE, ASSIGN, COUNT, COPY

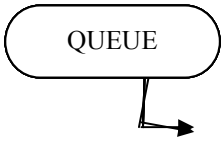


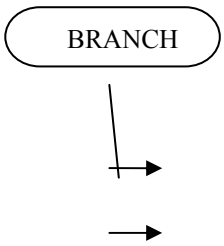
Bu üç blok türü çok fonksiyonlu bloklardır. Operation blok geniş ölçüde modelleme fonksiyonlarını içerir. Gelen bir entity daima bloğa girer, bazı fonksiyonları yerine getirir ve bu bloktan çıkar.

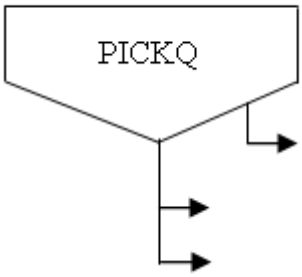
CREATE, CLOSE, ASSIGN, COUNT, COPY v.s. gibi bloklar "OPERATION" bloklardır.

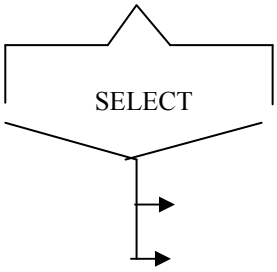
Hold blok gelen bir entity'i, sistemin mevcut durumunu göz önüne alarak, blok dışında tutmayı sağlar. Örneğin meşgul bir makinanın önünde bir entitiyi bekletir.

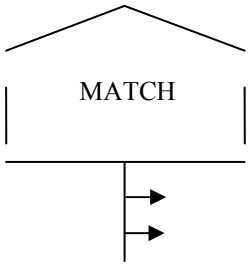
Bu blokta çok fonksiyonlu bir bloktur. Bir entitiyi bir yerden sistem içindeki diğer bir yere göndermeyi sağlar.

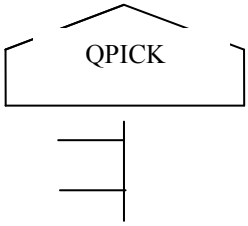












Bu 7 blok tek fonksiyonlu bloklardır.

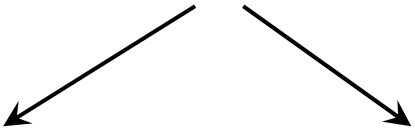
Operation Blokları

Entity'nin yaratılması
Entity'nin bekletilmesi
Entity'ye attribute atanması v.b. gibi fonksiyonların yerine getirilmesi için kullanılır.

BLOK OPERAND'leri

Blok operand'leri, bloğun işlemini kontrol eder, Örneğin bir DELAY bloğunda entity'nin ne kadar bekleyeceği operand ile belirlenir.

SIMAN'da iki tür sabit (constant) değer tanımlanır.



"Integer" constant
+ } kullanabilir.
-

İşaret yoksa pasitive demektir.
Örnek: -2
3 gibi

"Real" constant
a) – ınteger part.optional fraction
+

b) exponent olarakta girilebilir. (E ile)
Örneğin 10 sayısı
- 10.0
- 1.e1 veya
- 100.E-1 olarak girilebilir.

{ 2.0 } ile
2. aynıdır

VARIABLES (DEĞİŞKENLER)

Sistem bileşenlerini karakterize eden değişebilir değerler seti, SIMAN'da "variables" olarak geçer.

a) Special-Purpose variables: Özel-Amaçlı

SIMAN'da önceden tanımlanmış değişkendir. Model içinde otomatik update edilirler. Örneğin;
TNOW: Benzetim saatinin current (o anki) değeridir.

b) <u>General-Purpose Variables: Genel-Amaçlı</u>

SIMAN'da önceden tanımlanmamışlardır. Model içinde tanımlanır. Ve bir isme atanırlar. (sayı veya isim olabilir.)

n. değişken V(n) olarak tanımlanır.

Örneğin V(1), variable 1'in o anki değeridir. Variable adları harflerden, sayılardan ve special karakterlerden (-, ≠ (pound), @ (at sign), %, $ (dolar sign)) oluşabilir.

Örneğin ≠ 1, Data, 1_job geçerli kullanımlardır. 1E geçersiz kullanımlardır.

SIMAN'da büyük küçük harf kullanımı önemsizdir. DATA, Data, data aynı isimlerdir.

Bu; Variables, block adları ve diğer keywoed'ler içinde geçerlidir.

v(1), V(1) aynıdır.
DELAY, Delay, delay aynıdır.

<u>ONE OR TWO DIMENSIONAL ARRAY</u>

<u>ARRAY:</u>

Bir veya birden fazla değişken değerinin gruplandırılması yapılabilir.

Array adı Rejects olsun. 3 elemanı olsun.

Rejects (1), Rejects (2), Rejects (3),

İnteger olmalı

Matrix (3, 4) , 3. satır, 4. kalan değeri.

İnteger olmalı

Fortrandaki gibi

Attributes

Her bir entity'e ait onunla birlikte hareket eden attribute'ler vardır.

a) **Special-purpose attributes:** Önceden SIMAN içinde tanımlanmıştır. Bunlar ilerde inclenecek.
b) Bir özel proses modellenirken özel anlam içeren attributeler tanımlanabilir.

Örneğin bir fabrika modelinde her bir entity iş parçasını göstersin.

Genel amaçlı attribute'ler kullanarak, parça türünü, teslim tarihini belirleyebiliriz.

Bunlar bir sayı veya kullanıcı tarafından verilen bir isimle tanımlanabilir.

A (n) n. attribute değerini gösterir.
A (1) 1. attribute değerini gösterir.
A (2) 2. attribute değerini gösterir.

1. ve 2. attribute hangi özellik olarak kullanıldığını (user) bilecek.

Attribute'ler genel bir array adı kullanılarak gruplandırılabilir.

Mark Time (1) Mark Time (2) Mark Time (3)

Modelde kullanılacak genel-amaçlı attribute sayısı hakkında bir kısıt yoktur.

*blok operand'i olarak attribute kullanılabilir.

Örnek: DELAY

bekleme zamanını Process Time attribute ile tanımlayabiliriz. Bu durumda bekleme zamanı sabit değildir. Process Time tarafından belirlenir.

Random Variables

Bir çok proseste bir veya birden fazla rassal bileşen bulunur.

Örneğin işlem zamanı önceden tanımlanmış bir dağılıma uygun bir rassal değişken olabilir.
SIMAN'da yaygın olarak kullanılan dağılımlar fonksiyon olarak tanımlanmıştır ve bunları kullanarak r.d. üretebiliriz.
Figure 3.2 p.68

Expressions & Conditions

Bir veya birden fazla sabitten, attribute'den, değişkenden ve rassal değişkenden oluşan bir expression SIMAN'ın nümerik operand'ı olarak kullanabiliriz.

Expression'lar

- Standard aritnetik operatörler
  (+)(-) (*) (/) (**=exponentiation) ve parantezler ()

- sine, cosine, modulo, etc. (önceden tanımlanmış fonksiyonlar) dan oluşturulur.

DueDate / -4 geçersizdir.
DueDate / (-4) geçerlidir.

Condition'lar

TNOW < 40 or
TNOW.LT.40 Fortran form

.AND**.** **.**OR**.** kullanılarak conditon'lar birleştirilebilir.

Figure-33
s.70

SIMAN'da Expressionlar ve conditionlar aşağıdaki operatör önceliklerine göre değerlendirilir.

1. öncelikle ( ) ler
2. Aritmetik operatörler
   i. exponentiation (**)
   ii. (*, /)

iii. (+, -)

3. **.**LT**.** , **.**GT**.** , **.**EQ**.** , **.**NE**.** , **.**GE**.** , **.**LE**.**
   < , > , == , <> , >= , <=

4. Logical operatörler
   i. **.**AND**.**
      .OR**.**

Sembolik İsimler

Bazı blokların operand'leri sembolik name olarak tanımlanabilir.
Örneğin, SIMAN'da bir kuyruk tanımlayalım. Her bir kuyruğa bir sembolik name (buffer) atayalım ve bu adı blok operandi olarak o kuyruğu belirtmek için kullanabiliriz.

Model içinde kuyruklar ayrıca numaralandırılabilir. Ve bu kuyruk nosu aynı kuyruğun referansı olarak kullanılabilir.

Blok Label'ları

Herbir bloğa isteğe bağlı (optional) label atanabilir. Eğer model içinde bir yerden herhangi bir başka bloka gönderme yapılacaksa mutlaka o bloğun bir label'i olmalı.

Bloğun sol tarafında 'Label' tanımlanır. Label'ların uzunluğu rassaldır, duruma göre verilir.

Blok Commentleri: Blok fonksiyonunu tanımlamak amacıyla bir veya daha fazla blok commenti için er blokta özel alan vardır.
Bu alanı kullanmak optional dır. Ancak kullanımı tavsiye ediliyor.
Commentler her bir blok sembolünün sağına yazılır.

Blok Modifier: SIMAN'da tanımlı standart blok sembolleri vardır.Bunlar kullanılar bir blogun temel fonksiyonu genişletilebilir.veya modifiye edilebilir. Blogun sağına veya alt tarafın ayazılır.

Bloklarda Entity Akışı

Blok diyagramı modelin statik bileşenlerinden oluşur. Dinamik bileşen ise entitidir ve blok diyagramı boyunca bir bloktan bir bloğa hareket ederler.

Bloklar arasında bir entitinin akışı ardışık akış konnektörüyle ve NEXT ve DISPOSE modifierlarıyla kontrol edilir. Bunlar blogun çıkış tarafında yer alır. Bu semboller operation bloklarına eklenir.

Akış konnektörü: 

Çıkan entitileri sıradaki en yakın bloğa yönlendirir.

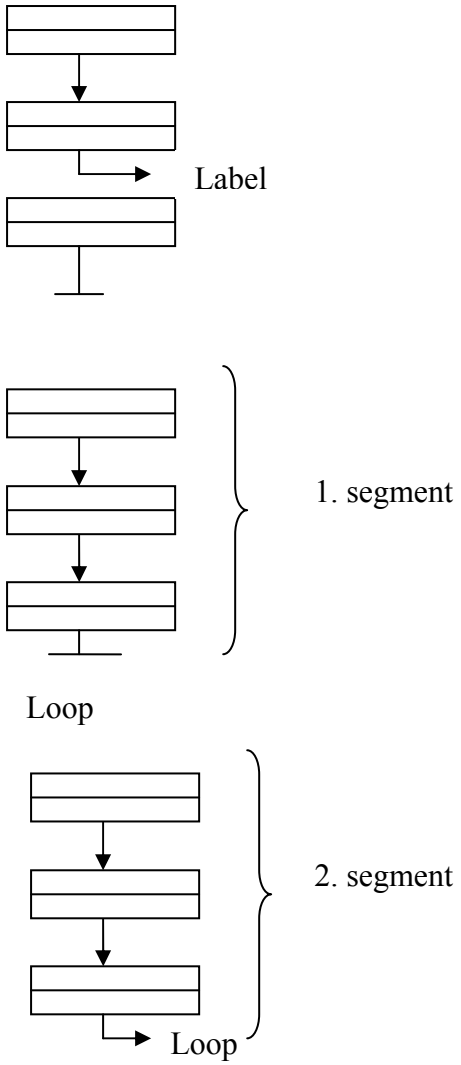
Next modifier: 

Çıkan entitileri operand Label'da tanımlı bloga gönderir.

DISPOSE Modifier: 

Bloktan çıkan entitileri modelden çıkarmak için kullanır.

Şekil 1'de akış konnektörü NEXT modifier ve DISPOSE modifier birlikte görünmektedir.



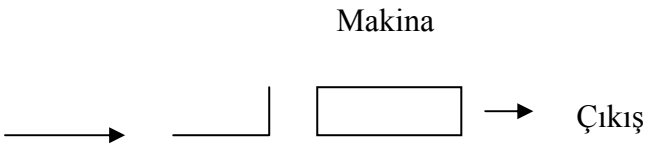
Şekil 2

Burada Block diyagramın 2 segmentten oluştuğu görülmektedir. 1. segment blok1'le başlar. DISPOSE Modifier olan blok 3'te biter

2. segment 4. blokla başlar ve 6. blokta biter. Blok 4'te gelen entiti 6. bloğa gelince Next modifier ile "loop" labeli ile belirlenen bloğa döner. Burada entiti sonsuz bir döngüye girer. Burada 1 ve 4 bloklar kaynak noktalardır. Entitiler buralardan modele girer. 3. blok batma(sonlanma) noktasıdır. Burada entitiler modelden çıkar.

Birçok sistemde birden fazla model segmenti kullanılabilir. Sınırı yoktur. Örneğin bir makinada işlemler bir segment içinde, makinanın arızalanması ve tamir işlemleri ise ayrı segmentler olarak programlanabilir.

Örnek:
İş parçalarının bir makinada işlem almak üzere sisteme geldiklerini ve işlemin yapılabilesi için sırada bekledikleri bir sistemi aşağıdaki gibi şematize edebiliriz.



Bir anda sisteme bir iş parçası girmekte ve bunların varışlararası zaman aralığı ortalama 4.4 dakika olan üstel dağılıma uymaktadır. Makinada gereken işlem süresi de rassal değişken olup üçgen dağılmaktadır. min=3.2 dak, mod=4.2 dak., max=5.2 dak.
Sistem tek vardiya olarak günde 8 saat çalışmakta ve hafta sonları çalışmamaktadır. Vardiya dışı ve hafta sonunda varış yoktur. Bir günü sonunda işlem sırasında kalan bir iş ertesi güne aktarılarak kesintisiz devam eder.
Bu sistemin performans ölçütleri olarak aşğıda verilen ölçütlerden bir ya da bir kaçı dikkate alınabilir.
Haftalık çıktı oranı(Bir hafta tanımlanan iş sayısı)
Makinanın doluluk oranı(Kullanım oranı)
Bir parçanın sistemde geçirdiği ortalama zaman
Bir parçanın sistemde geçirdiği en fazla zaman
İşlem görmek için bekleyen (kuyrukta bekleyen) ortalama iş sayısı

Öncelikle 1. performans ölçütü ile ilgileniriz.
Burada sisteme gelen işler modelde entity olarak modellenir.Sistemi tanımlamak için aşağıdaki adımlar kullanılır.

1- Sisteme gir
2- Kuyrukta bekle(makina boşalana dek)
3- Makineye gir
4- İşlem süresi kadar makinayı meşgul et
5- Makinayı serbest bırak
6- İşlem tamamlanan parça sayısını bir arttır ve sistemden çık

Bundan sonraki işlem Block Diyagramı çizmektedir.Önce gerekli Blokları tanımlayalım.

CREATE BLOK

Bu blok modelde entitilerin yaratılması işlevini görür. Yani önceden tanımlanmış bir varış prosesine uygun biçimde varışlar ardışık olarak programlanır.
CREATE blok bir entity kaynak noktasıdır ve bir operation bloktur.
Model segmentleri genellikle Create blokla başlar.
Bir model içinde hangi sayıda CREATE blok kullanılabileceği konusunda bir sınır yoktur.
Entity üretimini kontrol etmek üzere 4 operand vardır.

Create, Batchsize,offset
Interval, Maxbatches

CREATE, Batchsize,Offset: Interval, Maxbatches;
Batches: Varış sırasında sisteme giren entity sayısı. Default'ı 1'dir.
Offset:Simülasyonun başladığı andan ilk varışın yapılacağı ana kadar gecen zaman Default=0
Interval: Varışlararası zaman Default: infinity, yani 2. varış yok demektir.
MaxBatches: Varış sayısı. Default:infinity

Zaman birimi önceden tanımlı değildir
dak
saat
gün
veya diğer bir birim kullanılabilir.
Her yerde aynı birim kullanılmalıdır.

CREATE
EXPONENTIAL(30)

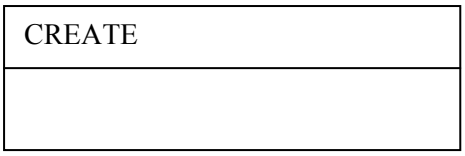
Bu kullanımda Batch size default=1 alınır.offset time=0
varış sayısı limiti=∞, varış prosesi ortalaması 30 dakika olan üstele uygun tarzda modellenir.

1. varış zamanı= EXPO(30), TNOW=0
2. varış zamanı TNOW= TNOW+EXP(30)
3. varış zamanı TNOW= TNOW+EXP(30)

İlk varış için offset üstelden zaman kadar dikkate alınır.

CREATE,EXPONENTIAL(30)
EXPONENTIAL(30)

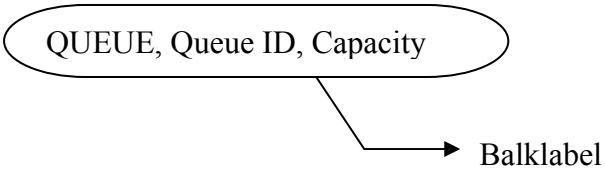
Bir blok sadece 1 kere entity üretir.(o entity de benzetimin başlangıcında üretilen entitydir.)



QUEUE Blok

Entityler için bekleme alanı oluşturulur.

QUEUE blok hold blokla birlikte kullanılarak status delay modellenir.Hernerde hold blok kullanılırsa bir önceki blok mutlaka hold blok olmalıdır.



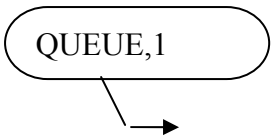
Entitiler kuyrukta bir sıralamaya girer. Kuyruktaki entity birinci sıradaki entitidir. Default sıralama FIFO'dur.
Experiment modelleme segmentinde sıralama yeniden tanımlanabilir. İlerki konularda öğrenilecek.

QUEUE blok 7 adet tek fonksiyonu bloktan biridir.SIMAN' da her kuyruk için bir sayı veya kullanıcı tarafından tanımlanan bir sembolik isim atanır.
Sayı ise 1'den başlar ve sırasıyla birer arttırılarak gider. Aynı modelde iki aynı isim farklı kuyruklara verilemez.

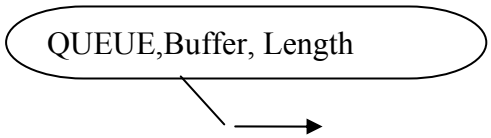
Default yoktur ve verilmesi gerekli bir operand dır.
Capacity kuyruk uzunluğunu göterir. Default=∞
Kuyruk limiti olduğunda, gelen entiti kuyruğa kabul edilmez. Balk label'da belirtilen alternatif bloka gönderilir.
Kuyruk limiti tanımlandığı halde balk label tanımlanmamışsa kuyruga girmeyen entiti sistemden çıkar.(tekrar geri dönmez)

NQ(QUEUE ID), kuyruk için tanımlanmış özel amaçlı değişkendir ve kuyrukta bekleyen entity sayısını verir.

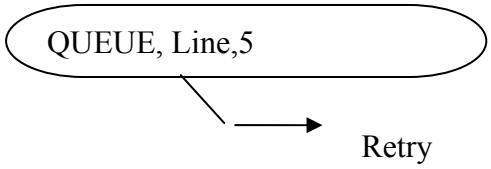
örnek: NQ(1)=3, QUEUE 1'de 3 entity var.
NQ(BUFFER)=5 Buffer adlı kuyrukta 5 entiy var.
SIMAN NQ( ) değerini otomatik update eder.



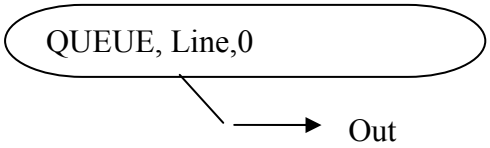
QUEUE ID=1 den infinite kapasiteli kuyruk



Buffer isimli kuyruk kapasitesi genel amaçlı length variable ile tanımlı.
Lenght'in o anki değeri kuyruk kapasitesini belirler ve bu değer benzetim içinde değişebilir . Burada Balk label tanımlı değil. Kuyruk dolduğunda gelen entity sistemden çıkar.



Line kuyruğu kapasitesi 5, Balk label= Retry



Kuyruk adı, Nospace, 0 kapasiteli
Entitylerin kuyrukta beklemelerine izin verilmez. Bu durumda takip ede Hold Blokta bir gecikme yaşamayanlar hariç, sisteme giren tüm entitiler out label'ında tanımlı bloğa yönlendirilir.

Kaynak Kullanımı: SIEZE BLOCK

Resource: Bir veya birden fazla özdeş objeler
Bir kaynak objesi bir entity tarafında kullanır. Benzer kaynak birimleri "kaynak kapasitesi" olarak tanımlanır.
Bir kaynak biriminin durumu "status" : boş veya doludur. Boş bir kaynak birimi bir entity tarından kullanıldığında status dolu konuma geçer. Entity'nin işlemi bittiğinde kaynak biriminin statusu otomatik olarak boş konuma geçer.
Yukarıda verilen örnekte makina bir birimlik kaynak olarak tanımlanacaktır.
SIMAN 'da Resources (kaynaklar) hem isimlendirilir hemde numaralandırılır.

NR (Resource ID) = Resource ID olarak tanılanmış kaynağın durumu
Resource ID, kaynak nosu veya bir isim olabilir.

MR (Resource ID) = modeldeki kaynak birimi sayısı

NR(1) veya } Bunların değeri 0 (idle) veya 1 (busy) olur.
NR(Machine)

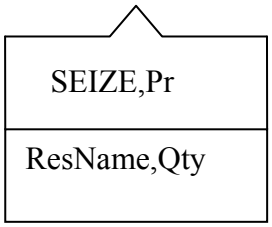
RESOURCE için no, isim, kapasite Experiment Frame de tanımlanır. Böylece block diyagram değiştirilmeden kapasite değişimi Experiment Frame de yapılır.
SEIZE Block bir entity' nin kaynağı kullanması işlemini yerine getiren bir bloktur.

SEIZE Block bir HOLD Block ve QUEUE blocktan sonra kullanımalıdır.

Pr: Priority (aynı kaynak için bekleyen entitynin önceliğini belirler)
ResName: Resource name (entity tarafından istenen kaynak ismi)
Qty: Unit sayısı (kaç unitin entity için tahsis edileceği)

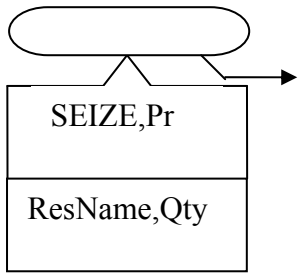
Default=1

SEIZE, Pr : ResName, Qty;
Repates;

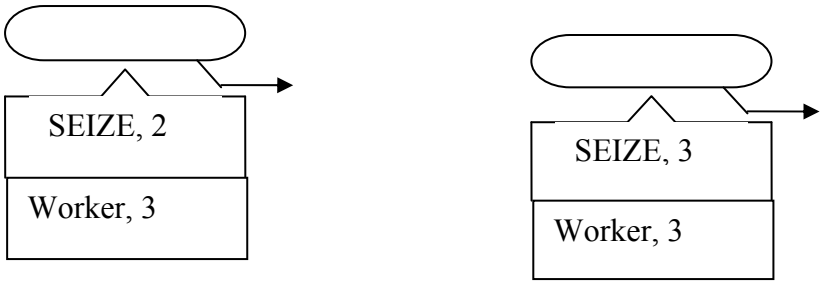


Bu problemde kaynak adı makina
Kullanılacak kaynak birim (unit) sayısı = 1
Eğer bir entity için Qty > 1 olarak tanımlanırsa, entity bu kaynakların tümü boşalana kadar bekler.
Kaynaklar QUEUE içinde bekleyen entitylere kuyruğun başından başlayarak hizmet verirler.
Örneğin kuyruğun başında bekleyen birinci entity iki makinada işlem görmek için bekliyorsa, ikinci entity sadece bir makinadan işlem alacaksa ikinci, birinci entity'nin her iki

makinadan işlem alıp kuyruktan çıkmasını beklemek zorundadır.
Birçok farklı kaynak aynı SEIZE block içinde kullanıma sunulacaksa, ikinci satırdaki ResName, Qty "repates" komutuyla her ilave aynak türü için kullanılır.
Aynı SEIZE block ile kullandırılacak farklı kaynak türleri için bir limit yoktur.

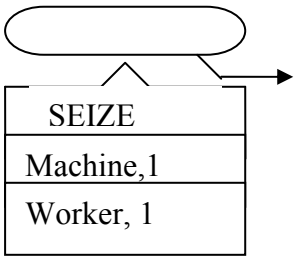


Operandları tanımlı olmayan bir SIEZE block.Görülüğü gibi QUEUE blockla birlikte kullanılıyor.
Entityler QUEUE-SEIZE block kombinasonuna gelirler. Sırasıyla sıralarını QUEUE içinde beklerler.
Bir entity QUEUE blocka girdiğinde bir makina boş ise dolu makina sayısı bir arttırılır ve entity beklemeden SEIZE blocka geçebilir.



Bu durumda aynı anda bu iki segmente gelen entitilerden 2 önceliğine sahip olana önce hizmet verilir.

Entityler Worker adlı kaynaktan 3 birim yakalamak (SEIZE) için beklerler. 3 kaynak irimi müsait olana dek hiçbir entity SEIZE blocka geçmez. Kaynağı kullanma önceliği "2" dir. Model içinde herhangi bir yerde tanımlı önceliği bir olan entity bu kaynakları o anda kullanmayacaksa bu entity kullanır.



Gelen entityler aynı anda hem Machine, hemde Worker adlı kaynaklardan birer birim kullanabilinceye kadar eklerler. Her iki kaynaktan birer tane kullanımı mümkün değilse herhangi birini kullanamazlar.

BEKLEME TANIMI: DELAY BLOCK

Herhangi bir kaynakta bir entitynin işlem süresi, makinenin hazırlık zamanı, muayene süresi gibi zaman işlemleri DELAY blokta yapılır.

DELAY: Duration, StorID;

Duration : Her bir entity için DELAY bloktan geçiş zamanını tanımlar.
T : Geliş zamanı
T+Duration : Çıkış zamanı

DELAY
Duration, StorID

StorID : Bir sayaç gibi kullanılır. NSTO(storID) değeri bir entity geldiğinde artar.
StorID içinde de kalış süresi tutulur.

Aynı anda belirli bir süre bekletilecek entity sayısı için bir kısıt yoktur.

StorID : Bu operand bir storage operandıdır. İki amaca hizmet ecen bir operandır. Storage facility bir alan tanımlar ki bu alan DELAY de tanımlı süre boyunca entitiyler tarafından kullanılır. DELAY bloktaki entitylerin sayılarına ilişkin istatistikleri toplayan bir mekanizmadır. Her storage facilitynin bir tanımlayıcısı vardır. Bunlar sayısal veya sembolik isimlerdir.

Eğer bir storage facility identifier DELAY blokta belirtilirse, bir entity DELAY bloğa geldiğinde SIMAN otomatik olarak NSTO(StorID) değerini arttırır, tersi durumda yani entity çıktığında ise azaltır. NSTO, SIMAN'da önceden tanımlanmış bir değişkendir ve bir storagedaki entity sayısını tutar.
Aynı storage facility iki veya daha fazla DELAY blokta kullanılırsa, o zaman NSTO(StorID) değeri bu storage facility için atanmış tüm DELAY blokların içindeki entity sayısına eşit olacaktır.
StorID'nin ikinci amacı ise animasyonla ilgilidir. CINEMA animasyonu için bir arayüz görevi üstlenir. Entitylerin DELAY bloktan geçiş animasyonunu sağlar.

Örnek:

DELAY
Setup+Process

Bu bloğa giren entity hazırlık ve proses attributelerin değerleri toplamı kadar bekler. Bu
bloktan geçen entityler storage facilitye atanmaz.

DELAY
10, Stor2

Bu bloğa giren entityler 10 dakika beklerler. Bu süre boyunca entityler storage facility 2'yi kullanılırlar ve bu süre store2 içinde tutulur.

KAYNAĞI SERBEST BIRAKMA (BOŞALTMA) : RELEASE BLOCK

RELEASE
ResName, Qty

RELEASE
Machine

RELEASE : ResName : Boşalacak blok adı
Qty : Birim sayısı Default=1
Repeats;

Çıkan entity Machine adlı serverdan 1 birimlik yer boşaltır.

RELEASE
Nurse, 2
Doctor, 1

Her gelen entity çıkarken 2 hemşire ve 1 doktoru serbest bırakır.

SAYMA İŞLEMİ : COUNT BLOCK

Bu blok benzetim programının terminasyonu için kullanılır. SIMAN sayma işlemi için counter (sayaç) kullanır.

CounterID: sayı veya sembolik isim olabilir.

Bir counter'a bir veya birden fazla COUNT bloktan atıf yapılabilir. Counter'ın bir limiti vardır. Benzetim programı bu limitin kontrolü ile durdurulabilir.

COUNT: CounterID, Increment

Increment: Artış değeri, Default =1

COUNT
CounterID, Increment

NC(CounterID) değişkeni, SIMAN'da CounterID ile belirlenen counter'ın o anki değerini verir.

Örnek:

NC(JobsDone) : JobsDone adlı counter'ın o anki değeri
NC(7) : Counter7'nin o anki değeri

COUNT
JobsDone

JobsDone her entity girişi ile 1 artar.

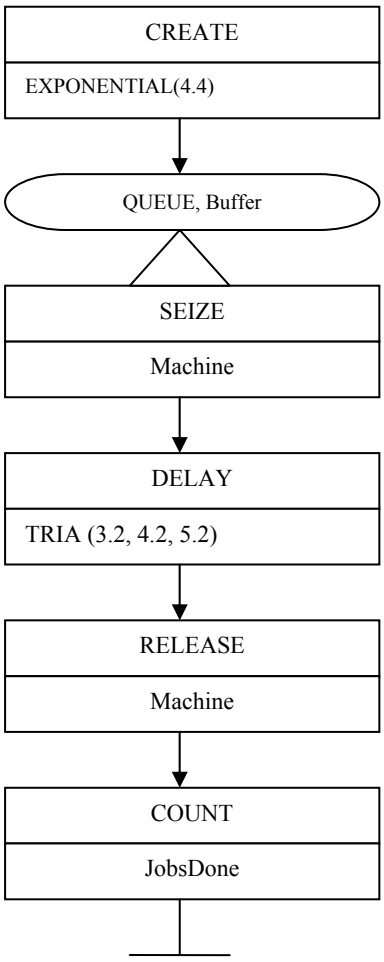
COUNT
7, NumBatch

Counter 7, her seferinde NumBatch attribute değeri kadar arttırılır.

Bu bloklar içindeki değer counter limit değerleriyle karşılaştırılır ve benzetim durdurulur veya devam eder.

BLOCK DİYAGRAMI

```
BEGIN;
      PROJECT,          Sample Problem 3.1, SM;
      DISCRETE,         100;
      QUEUES:           Buffer;
      RESOURCES:        Machine;
      COUNTERS:         JobsDone;
      REPLICATE,        1, 0, 480
END;
```



Basit problemin blok diyagramı.

- Entity modele CREATE blokta girer.

- Varışlar arası zaman aralığı ortalaması 4.4 olan üstel dağılıma uygundur.
- Seize ile, 1 makine server olarak kullanılır.
- Boş makine yoksa QUEUE'da bekler. (QUEUE'nın adı Buffer)
- Delay'de setup+işlem zamanı tanımlanmıştır. Bu zaman üçgen dağılıma uygundur.
  min= 3.2, mod= 4.2, max=5.2
- Bu zaman sonunda entity RELEASE bloğa geçer.
- 1 makine boşlatılır.
- Entity Count bloğa geçer, JobsDone 1 arttırılır.
- Entity modelden çıkar.
- DISPOSE

SIMAN'da, block sembolleri ile statementların bire bir karşılığı vardır.
Block statementına ilave olarak BEGIN ve END kullanılır.

BEGIN, Listing; Programın ilk statementıdır.
Listing, modelin işletilmesi sırasında model üretilmesini kontrol eder. Default= Yes

Eğer YES ise, model statementları ekranda görülür.
NO ise, model statementları üretilmez, compiling hızlanır.

Genellikle BEGIN; kullanırız. (Default=YES olur)

END; en son blok statementıdır. Operandı yoktur. END; olarak kullanılır.

Block Statement Format

- Statementler için bir format yoktur. (Free format)
- Her input line 74 karakterle sınırlıdır.

: segment sonlandırıcısı
; statement sonlandırcısı

SAMPLE MODEL SOURCE FILE

```
BEGIN;
  CREATE  : EXPONENTIAL (4.4);
  QUEUE,    Buffer;
  SEIZE     :  Machine;
  DELAY   : TRIANGULAR(3.2, 4.2, 5.2)
  RELEASE: Machine;
  COUNT   : JobsDone: DISPOSE;
END;
```

DENEYSEL ELEMANLAR (Experimental Elements)

PROJECT ELEMANI

Project, Title, Analyst, Date;

Title : Proje adı } Alfanümerik isim
Analyst: Analist adı
Date : Ay/Gün/Yıl veya Gün/Ay/Yıl

- Her bir isim 24 karakteri geçemez.
- 24'den fazla kullanıldığında fazlası atılır.

Örnek: PROJECT, Example 3.1, SM;

Tarih Default olarak host computer'dan okunan tarih olarak gelir.
Bu eleman kullanılmazsa çıktı dosyası vermez.

ENTITY SAYISINI SINIRLAMA : DISCRETE

Modelde mevcut entity sayısını sınırlı tutmak istediğimizde kullanılır. Eğer model çalışırken bu sınır geçilirse model durur ve "entity sayısı aşıldı" mesajını verir.

DISCRETE, Entities;

Örnek: DISCRETE, 100;

Bu durumda bir entity modele hali hazırda sistemde 100 entity varsa, benzetim şu mesajla sonlanır: "Entity limiti aşıldı", "Limit on the # of entities was exceeded".

KUYRUK ELEMANLARI

QUEUES: Number, Name, Ranking:
repeats;

Number : kuyruk numarası
Name : kuyruk adı
Ranking: Default= {FIFO}/{LIFO}/{LVF}/{HVF}

Number ve Name, SIMAN variable'ı NQ'ya argüman olarak kullanılır. (NQ=kuyrukta bekleyen entity sayısı). Number'ı

kullanmadığımızda SIMAN otomatik olarak sırasıyla 1'den modeldeki kuyruk sayısına kadar kuyruk numarası atar.

Ranking;

Low-Value-First-rule (LVF(AttributeID))

Bu kural kuyruğun belirlenen atrribute'un artan değerine göre sıralanmasına sebep olur.

High-Value-First-rule (HVF(AttributeID))

Bu kural kuyruğun belirlenen atrribute'un azalan değerine göre sıralanmasına sebep olur.

Örnek: LVF(1) : attribute 1'e dayalı low-value-first ranking belirtir.
HVF(Priority): priority adlı attribute'e dayalı high-value-first ranking belirtir.

Örnek: QUEUES: Buffer; Adı buffer olan bir kuyruk var ve FIFO ranking

<u>RESOURCES</u>

Bir veya birden fazla resources modelde kullanılacaksa " " experimentte olmalıdır.

RESOURCES: Number, Name, Capacity:
repeats;

Number : 1'den başlar.
Name : kullanıcı tarafından verilir. İleride SEIZE, RELEASE blokta kullanılır. Default'u yoktur.
Capacity: Default=1
Benzer ve birbirinin yerine geçebilen başlangıçta sistemde var olduğu tanımlanan kaynak kapasitesi.

Örnek: RESOURCES: Machine, 1; (1 kaynak tanımı, adı: machine, başlangıç kapasitesi=1)

COUNTERS

COUNTERS: Number, Name, Limit, InitOpt, OutFile: repeats;

Number : Counterslara numara verilir. 1'den başlar ve modelde kullanılan Counter sayısına ulaşana kadar verilir veya atlanır, Experiment processor otomatik olarak verir.
Name : Sembolik isim, counters adı, SIMAN özet raporda counter çıktı labelı olarak kullanılır.
Limit : Benzetim uzunluğunu kontrol etmek için tanımlanan entity limiti. Default=Infinity
InitOpt : InitOpt (reinitialization) birden fazla replication (deneme) yapılacağı zaman sayaçların başlangıç değerlerini kontrol eder. Default=YES 'tir, bu da her bir

deneme başında counter değerinin ‘0’ lanmasını sağlar.
OutFile : Opsiyoneldir. Eğer tanımlanırsa SIMAN counters değerinin tüm zaman içindeki
değerini tutar.

Örnek: COUNTERS, JobsDone:
Rejects, 20, NO, “C:\Rejects.dat”;

Bu eleman iki counter tanımlar:

1. JobsDone , limiti ve file tanımı yok
2. Rejects , 20 limit, denemeler arasında yeniden “0” lanmıyor ve özel bir çıktı dosyası var. C:\Rejects.dat

REPLICATE

İstenirse, Experiment frame’de “REPLICATE” kullanılarak simülasyon uzunluğu ve başlangıç opsiyonu olarak kullanılır.

REPLICATE, NumReps, BeginTime, Length, InitSys, InitStats, WarmUp

NumReps : Ardışık yapılacak deneme (tekrar) sayısı, tamsayı
BeginTime : İlk denemenin başlangıç zamanı olarak ‘TNOW’ a verilen başlangıç değeri
Length : Her bir denemenin maksimum uzunluğu. Eğer simülasyon başka bir şekilde
sonlanmamışsa TNOW length’de belirtilen maksimum değere ulaşınca
otomatik sonlanır.
InitStats : YES/NO, başlangıç değeri

InitSys ve InitStats, simülasyon denemeleri arasında değişkenlere başlangıç değeri verme işlemlerini kontrol eder.

- Başlangıç değeri alması isteniyorsa "YES", istenmiyorsa "NO" kullanılır. Default=YES
- Böylece her deneme için başlangıç değerlerine dönülür.
- Her bir denemede farklı rassal sayı seedleri kullanılacağından he deneme farklı sonuçlar verir.
- Eğer başlangıç değerleri verme işlemi yapılmazsa, örneğin 2. denemenin başlangıcı 1. denemenin sonuç değerleri olur, yani 2. deneme 1. denemenin devamı olur.
- Eğer istatistiklere de başlangıç değeri her denemede verilirse, bir deneme sonunda "0" lanır ve yeni deneme için başlangıç halini alır, sonuç rapor o denemenin sonuçlarına dayalı hazırlanır.
- Eğer istatistikler sıfırlanmayıp 2. denemede 1. denemede kaldığı yerden devam ederse 2. deneme sonunda elde edilen rapor 1. denemenin de istatistiklerini kapsar.

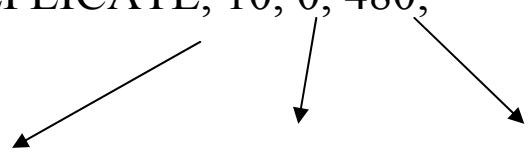
WarmUp

Yanlı gözlemleri, sonuçta elde edilecek performans ölçütlerini etkilemesin diye dikkate almayız. SIMAN'da bu işlemi "WarmUp" operandı yapar.

WarmUp için "YES" kullanılırsa her denemede geçici durum periyodu kadar deneme azaltılarak sonuç raporlar hazırlanır.

Eğer "NO" kullanılırsa, bir kere ilk denemeden önce uygulanır, diğer denemelerde uzunluk sabit kalır.

ex: REPLICATE, 10, 0, 480;



10 adet deneme	ilk deneme 0 zamanında başlayacak	Benzetim uzunluğu

Diğer 9 deneme aynı başlangıç değerlerinde başlayacak, tüm istatistikler temizlenecek,

Bu 9 deneme rassal sayı seedi hariç özdeş ortamda tekrarlanan özdeş denemeler olacaktır.

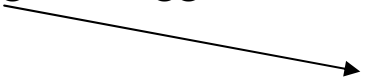
EXPERIMENT file'ının hazırlanması

BEGIN'le başlar

END'le biter

Experiment source file'da BEGIN ilk statementtır.

BEGIN, Listing, Debugger;



Interaktif debugger (hata bulucu) ları kontrol eder.

Listing "YES / NO" dur. Experiment list üretimini kontrol eder. Default: YES

'NO' kullanılırsa EXPERIMENT list üretilmez.

Execution daha hızlı olur.

Burada BEGIN'in operandlerinin default larını kullanacağız.

BEGIN; (Default'ları kullanacağımız için sadece BEGIN yeterli)

Ex:

```
BEGIN;
    PROJECT,        Sample Problem 3.1, SM;
    DISCRETE,       100;
    QUEUES:         Buffer;
    RESOURCES:      Machine;
    COUNTERS:       JobsDone;
    REPLICATE,      1, 0, 480
END;
```

Soru: Bu program ne zaman hangi koşullarda durur ?

```
BEGIN;
    CREATE:                         EXPONENTIAL (4.4);
(sisteme gir)
    QUEUE,          Buffer;                 (makina için
bekle)
    SEIZE:          Machine;
    DELAY:          TRIANGULAR (3.2, 4.2, 5.2);
    RELEASE:        Machine;
    COUNT:                          JobsDone: DISPOSE;
(Biten işleri say)
END;
```

COMPILING the MODEL ve EXPERIMENT file :

Model file adı Example.mod olsun.
Experiment file adı Example.exp olsun.

- SIMAN iki ayrı compiler'ı vardır. Biri model için diğeri experiment için
- Mod ve exp uzantıları kullanılır. (.m ve .e de kullanılabilir)

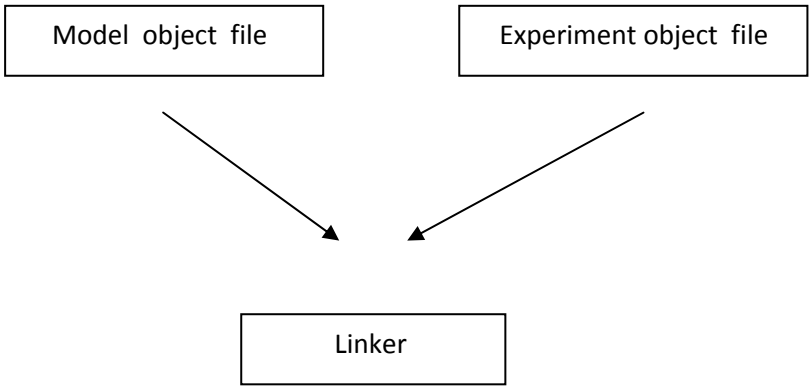
MODEL Example.mod
EXPMT Example.exp

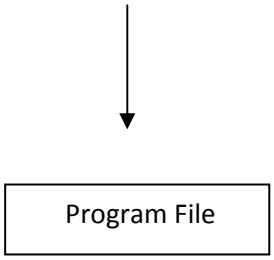
Model veya EXPMT compile işlemi sırasında SIMAN source file listesini ekrana gönderir. Bu listede SIMAN'ın her statementa otomatik olarak verdiği numaralar görülmektedir.
Bu listede varsa "error" mesajları bulunur.
Bu hatalar düzeltildikten sonra "compilation" işlemi tekrarlanır.

MODEL VE EXPERIMENT BAĞLANTISI

Model ve Exp. file'ları hatasız compile edildikten sonra, bunların program içinde bağlantılı olması sağlanmalıdır.

Linking İşlemi:





File'lar LINKER adlı SIMAN programının çalıştırılması ile 'Link' edilir.

LINKER, model ve experiment file'ları input olarak okur ve program file'ını output olarak yazar.

Bizim örnekte,

Example.m ve Example.e (file isimleri)

Bizim program file'mızı Example.p olarak çağırır.

LINKER Example.m Example.e Example.p

Herbir source file errorsüz compile etmemize rağmen model ve experiment file'ları arasındaki uyumsuzluk nedeniyle hala hata "error" olabilir.

Örneğin, model bir kaynağı referans göstermiş ancak o kaynak experimentte tanımlanmamış olabilir. Model ve exp. arasında bu tür uyumsuzluklar olduğunda LINKER program onları belirler. Bunlar "Link-time-errors" olarak adlandırılır.

Böyle bir error varsa, model düzeltilir ve modifiye edilmiş file'lar yeniden compile edilir ve link adımları yeniden tekrarlanır.

EXECUTING

File'lar başarılı bir şekilde link edildikten sonra benzetim executing için hazırdır.

SIMAN Example.p

Compile ve link işlemleri sırasında hata olmamasına rağmen executing sırasında hata mesajı verebilir.

Bunlar "run-time-errors" olarak adlandırılır.

Bir "run-time-error" ortaya çıktığında benzetim sona erer ve SIMAN error mesajlarını gösterir.

"run-time-error" lerin bulunması ve düzeltilmesi "compile-time-error" lere ve "link-time-error" lere göre daha zordur.

Eğer benzetim koşumu hatasız sona ererse, SIMAN Summary Report görünür.

Aşağıda örnek problemin summary reportu görülmektedir.

Project: Sample Problem 3.1