X-GPSS

Víctor Garcia Carrasco; victor.garcia.carrasco@upc.edu

TO DO

- 1 select the GPSS version you prefer
- Evaluate the alternatives.

- Review a paper to understand how to select the appropriate software.

GPSS

- General Purpose Simulation System.
- Developed by Geoffrey Gordon during 60's of XX century.
- Discrete systems modeling / Discrete Event Simulation

GPSS world

- Entities (transactions) traveling through the system.
- Through the blocs.
  - The number of blocs is different depending on the GPSS version used.
  - Minuteman
  - **aGPSS**
  - **GPSS World**
  - gpss.py (https://martendo.github.io/gpss.py/web)
  - JGPSS
  - GPSS/H
  - …

Architecture

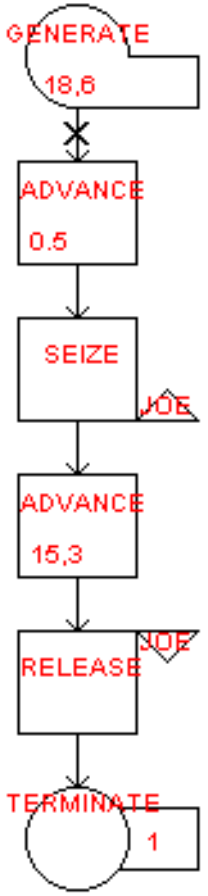
- Based in blocs diagrams.
- Blocs joined using lines representing a transactions sets, that makes its movement through the blocs.
- Entities making its path through the system elements. Transactions.
- Its movement is from bloc to bloc → representing actions or events that affects the entities.

Transactions

- Temporal or permanent.
  - Temporal: created and destroyed.
  - Permanents: dynamic.
- Have attributes.
- Individual and unique identifier.

GPSS code example

```
      SIMULATE
*
*     ONE-LINE, SINGLE-SERVER QUEUEING MODEL
*
      GENERATE      18,6      ARRIVALS EVERY 18 ± 6 MINUTES
      ADVANCE       0.5       HANG UP COAT
      SEIZE         JOE       CAPTURE THE BARBER
      ADVANCE       15,3      HAIRCUT TAKES 15 ± 3 MINUTES
      RELEASE       JOE       FREE THE BARBER
      TERMINATE     1         EXIT THE SHOP
*
      START         100 ( = TC , transaction counter)
      END
```

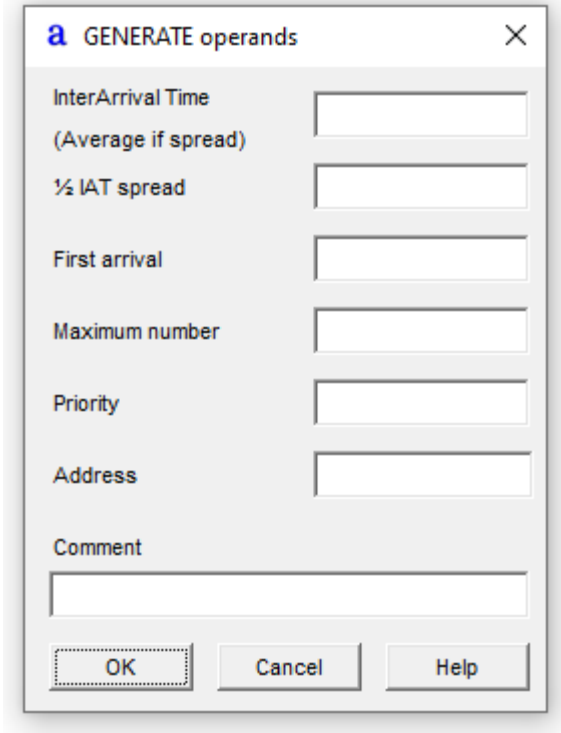


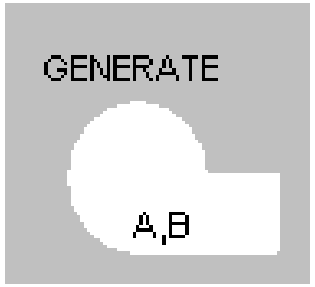
Blocs

Program logic instructions

Generate

- Creation of model transactions.
- Time between arrivals: random variable.
- A: Average interval time.
- B: ½ range (A ± B).
- C: Time for the first transaction.
- D: Maximum number of created transactions.
- E: Priority level





Terminate

- To destroy the transactions.
- A: Number to decrement the TC.

Advance

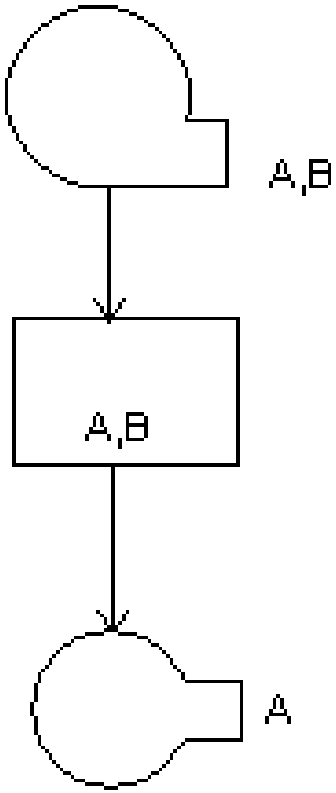
- Stops the transaction movement some time.
- A: Average waiting time
- B: ½ range

Example

- Museum

GENERATE 10,5

ADVANCE 20,5

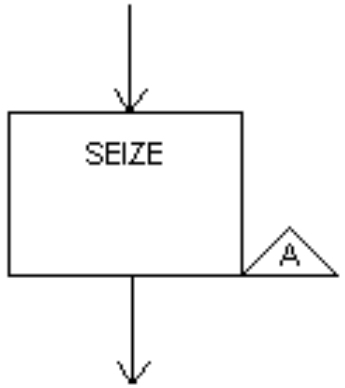


Modeling simple servers

- People or objects that performs a service.
- Limited resource-
- Kind:
  - Simple → 1 server by time unit.
  - Complex → more than one server by time unit.

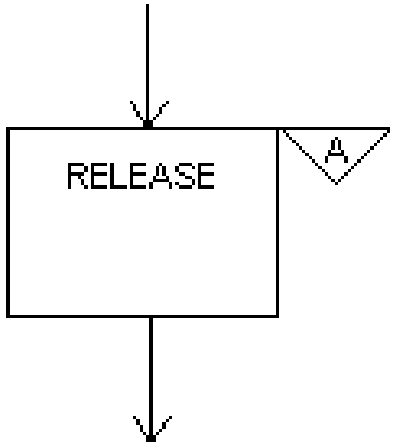
Seize

- The entity request the server.
- A: Identifier of the requested server.



Release

- To release a server.
- A: Identifier of the released server.



Example: Manual lathe

A manual lathe process wooden pieces with a 5 ± 2 minutes (uniform distribution). The arrival of the pieces follows a uniform distribution of parameters 7 ± 3 minutes. Develop a GPSS model to simulate the process of 500 pieces.

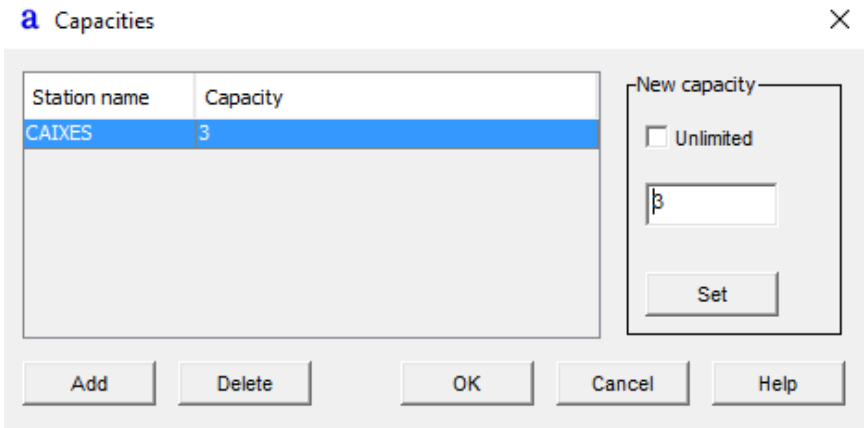
- Pieces arrival: 7 ± 3 (uniform, minutes)
- Time to process a piece: 5 ± 2 (uniform, minutes).

Example: Manual lathe (answer)

- GENERATE 7,3
- SEIZE TORN
- ADVANCE 5,2
- RELEASE TORN
- TERMINATE 1

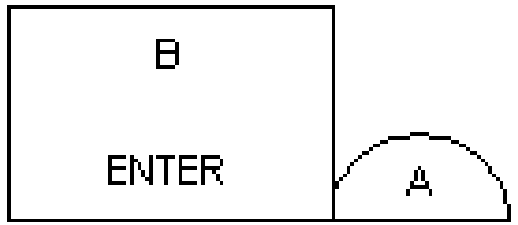
Modeling complex servers

- Is needed to define the server capacity.
- STORAGE S(ELEVATOR),6 (H)
- ELEVATOR STORAGE 6 (w)
- Is needed to show when the server is requested and when the server is released.
- Via Control -> Capacities (g)



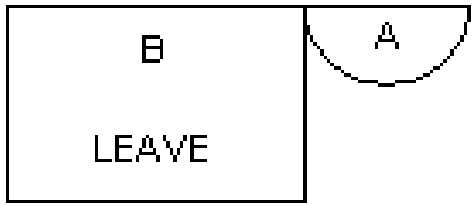
Enter

- Request of one ore more parallel servers.
- Simulates the enter of the entity in the server.
- A: server's name.
- B: number of servers requested.



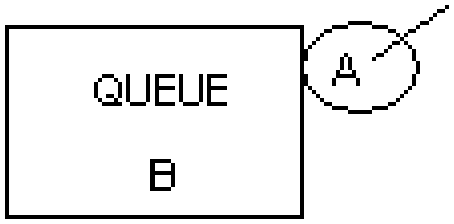
Leave

- To simulate the release of one or more servers.
- A: server's name.
- B: number of servers to release.



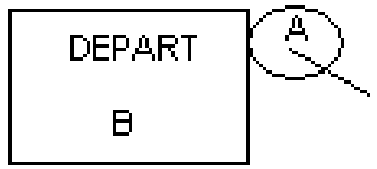
Arrive (QUEUE on [W])

- To model the queues in front of a server. It measures the time it takes for a transaction to go from the ARRIVE block to the corresponding DEPART block.
  - A: queue identifier.
  - B: Number of entities [W].



Depart

- To show that an entity is leaving a queue.
  - A: queue identifier.
  - B: number.



Example: Banc Fortuna v1.0

- In a banc the clients arrives following a uniform distribution of 5 to 9 minutes.
- 1 single cashier.
- Service time of 2 a 6 minutes, following a uniform distribution.
- Simulate 500 clients.

- Remember: we want QUEUE information.

Example: Banc Fortuna v1.0 (answer)

- GENERATE 7,2
- QUEUE CUA
- SEIZE CAIXER
- DEPART CUA
- ADVANCE 4,2
- RELEASE CAIXER
- TERMINATE 1

Here the focus is on pure waiting times on a specific resource

Example: Banc Fortuna v1.1 (answer)

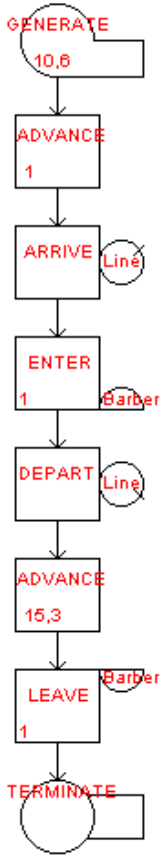
- GENERATE 7,2
- QUEUE CUA
- SEIZE CAIXER
- ADVANCE 4,2
- RELEASE CAIXER
- DEPART CUA
- TERMINATE 1

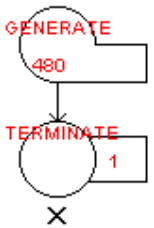
Here the focus is on the time the user stays in the system

Exercise: Barber v2

- Try with GENERATE 10,6 -> System overloads
- Put 2 Barbers -> Workloads and wait times are back to optimal levels.
- Simulates 8 hours of work with a timer

```
; Customer
        GENERATE    10,6        ; Customers arrive
        ADVANCE     0.5         ; Time to hang coat
        QUEUE       Line        ; Enter the line
        ENTER       Barbers,1   ; Capture a barber
        DEPART      Line        ; Leave the line
        ADVANCE     15,3        ; Use the barber
        LEAVE       Barbers,1   ; Free the barber
        TERMINATE   1           ; Leave the shop

; Timer
        GENERATE    480         ; Define a timer set time 480 (8 hours)
        TERMINATE   1           ; Shut off the run
; Control
Barbers STORAGE     2           ; Define 2 barbers
        START       1           ; Start the run
```





Storage	(1) Capacity	(2) Average contents	(3) Average utilization	(4) Entries	(5) Average time/unit
BARBER	2	1.48	73.78	48	14.76

Storage	(6) Current contents	(7) Maximum contents
BARBER	1	2

AD set	(1) Maximum contents	(2) Average contents	(3) Total entries	(4) Zero entries	(5) Percent zeros
LINE	1	0.07	48	36	75.00

AD set	(6) Average time/trans	(7) $Average time/trans	(8) Current contents
LINE	0.69	2.75	0

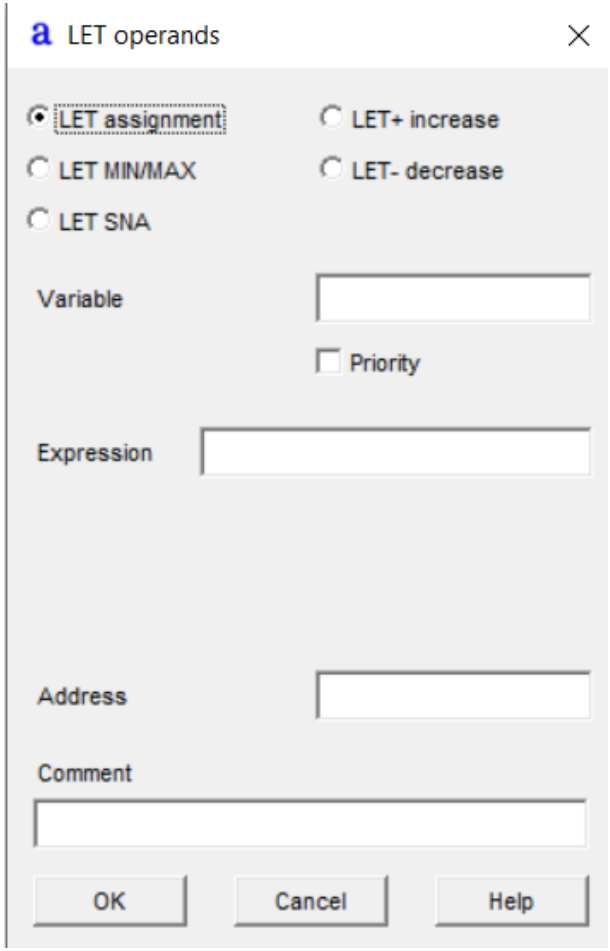
$Average time/trans=average time/trans excluding zero entries

Assign or LET

- Allows the modification of the transaction parameters.
- A: parameter's number.
- B: value to assign.

  - ASSIGN COLOR,4
  - ASSIGN TYPE,10
  - ASSIGN TIME,7.5
  - ASSIGN 1+,10

- P$COLOR To access the attribute
- ADVANCE P$COLOR



Labels

- Is allowed to name the GPSS blocs.
  - To access the SNA's (standard numerical attributes).
  - To break the transaction sequence.

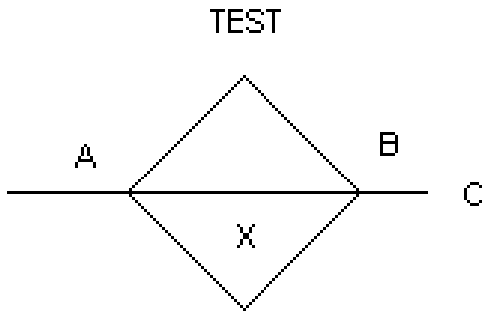
In aGPSS, the names are limited to a minimum of 3 and a maximum of 6 characters. The first three must be letters.

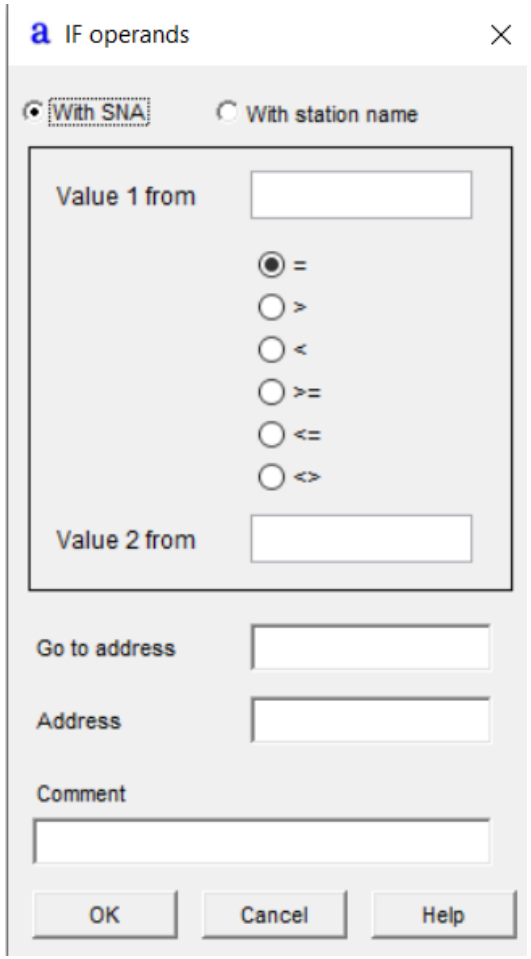
SNA's

- Provide information related to the model entities (sensors).
- Can be used in simulation time.
- Give information about the simulated model.
- Examples:
  - C1 or CL: Clock (clock simulation value, relative or absolute)
  - N$label : number of entries on block "label"
  - W$label: number of transactions on block "label"
  - R$name: remaining capacity on storage "name"
  - S$name: seized capacity on storage "name"
  - F$name: 1 facility "name" in use, 0 not in use
  - SF$name: 1 -> storage "name" is full, 0 is not full (GPSS World)
  - Q$name: number of transactions on queue "name"
  - M1: Transaction time of the current transaction, the absolute clock value minus the 'mark time' of the transaction. (GPSS World)

IF (Test on [W])

- Allows compare values and control the destination of a transaction.
- X: relation operator.
- A: verification operator.
- B: Reference value.
- C: number of the destination bloc.





Test (W)

- If the operand C is not defined, TEST is working in conditional mode. The transaction enters in the bloc and, when the condition is true, continues its movement.
- If C is specified, when the condition if false the transaction jumps to C.
- Values for X:
  - E: equal
  - G: greater
  - GE: greater or equal.
  - L: less
  - LE: less or equal.
  - NE: non equal.

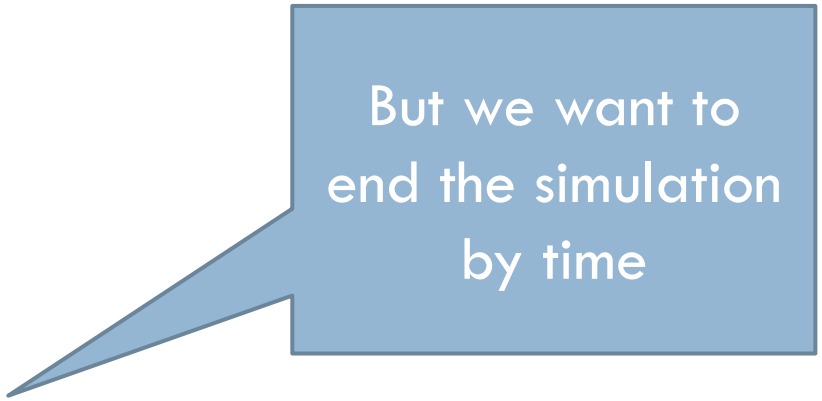
ASSIGN COLOR,4 TEST E P$COLOR,4,END **ADVANCE 10** END TERMINATE 1	ASSIGN COLOR,5 TEST E P$COLOR,4,END ADVANCE 10 **END TERMINATE 1**	ASSIGN COLOR,5 **TEST E P$COLOR,4** ADVANCE 10 END TERMINATE **1**

Example: Banc Fortuna V3.0

- In a banc the clients arrive following a uniform distribution with parameters 5 to 10 (minutes).
- 3 tellers.
- Service time: 2 to 5 minutes (uniform distribution).
- Simulate 1 day of work (4 hours open).
- At the end of the day no client must remain in the banc.

Example: Banc Fortuna V3.0 (answer)

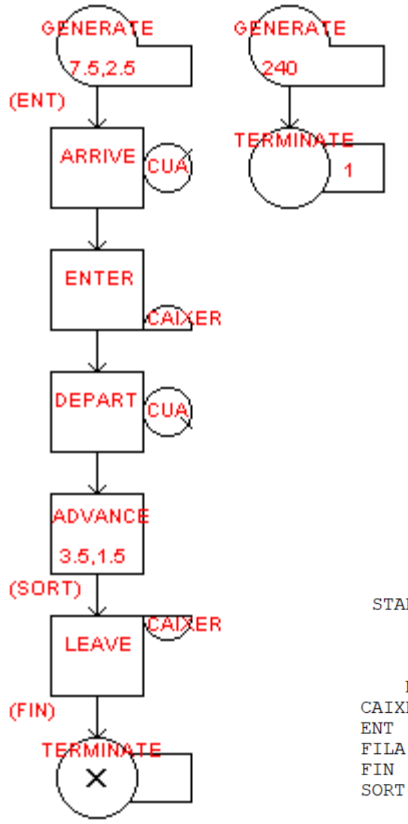
```
CAIXES STORAGE 3

            GENERATE        7.5,2.5
ENT         QUEUE   FILA
            ENTER   CAIXES
            DEPART  FILA
            ADVANCE         3.5,1.5
SORT        LEAVE   CAIXES
FIN         TERMINATE       500
```



Example: Banc Fortuna V3.0 (answer)

```
CAIXES STORAGE 3

             GENERATE 7.5,2.5
ENT          QUEUE    FILA
             ENTER    CAIXES
             DEPART   FILA
             ADVANCE  3.5,1.5
SORT         LEAVE    CAIXES
FIN          TERMINATE

             GENERATE 240
             TERMINATE 1

START                 1
END
```

```
SIMULATE 1

CAIXER CAPACITY 3
             GENERATE 7.5,2.5
ENT          ARRIVE CUA
             ENTER CAIXER
             DEPART CUA
             ADVANCE 3.5,1.5
SORT         LEAVE CAIXER
FIN          TERMINATE

             GENERATE 240
             TERMINATE 1

START 1
END
```



Friday, November 19, 2021 10:55:27

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	240.000	9	0	1

NAME	VALUE
CAIXES	10000.000
ENT	2.000
FILA	10001.000
FIN	7.000
SORT	6.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	32	0	0
ENT	2	QUEUE	32	0	0
	3	ENTER	32	0	0
	4	DEPART	32	0	0
	5	ADVANCE	32	1	0
SORT	6	LEAVE	31	0	0
FIN	7	TERMINATE	31	0	0
	8	GENERATE	1	0	0
	9	TERMINATE	1	0	0

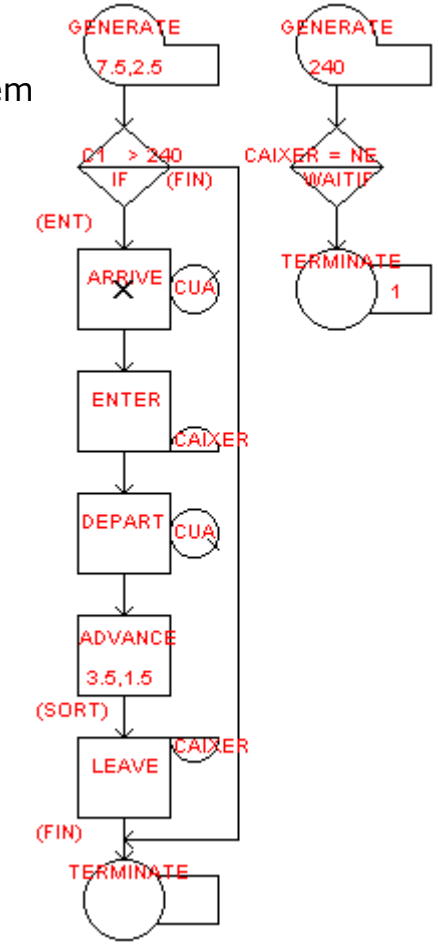
Example: Banc Fortuna V3.0 (answer)

```
CAIXES STORAGE 3

             GENERATE   7.5,2.5
             TEST LE    C1,240,FIN
ENT          QUEUE      FILA
             ENTER      CAIXES
             DEPART     FILA
             ADVANCE    3.5,1.5
SORT         LEAVE      CAIXES
FIN          TERMINATE

             GENERATE   240
             TEST E     N$ENT,N$SORT
             TERMINATE  1

START                   1
END
```

If simulation clock > 240, does not allow any entity to enter the system

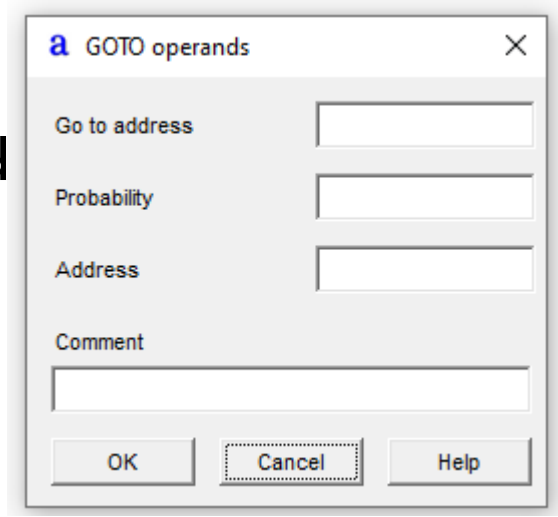
The transaction is blocked if the condition is not satisfied allowing all entities to exit the system

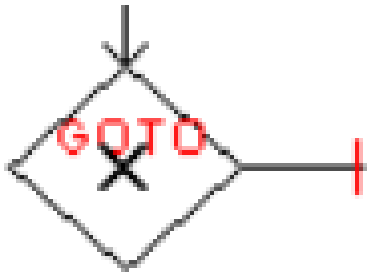
LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	32	0	0
	2	TEST	32	0	0
ENT	3	QUEUE	32	0	0
	4	ENTER	32	0	0
	5	DEPART	32	0	0
	6	ADVANCE	32	0	0
SORT	7	LEAVE	32	0	0
FIN	8	TERMINATE	32	0	0
	9	GENERATE	1	0	0
	10	TEST	1	0	0
	11	TERMINATE	1	0	0

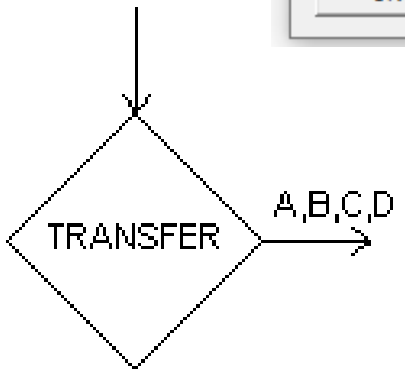


GOTO (Transfer)

- Allows to break the sequential movement of a transaction.
  - A: Probability of going to the indicated direction.
  - If there is no operator, then it is an unconditional GOTO and always sends the transaction to address.







Transfer [W]

- A:tranference modality
  - Both, All, Pick, FN, P, SBR, SIM, Fraction, Number, SNA, Null
- Optional parameter.
- B: number or bloc position.
- C: number or bloc position.
- D: number or bloc position.

Transfer

- TRANSFER .40,OPC1,OPC2
  - TRANSFER ,OPC1 ←GOTO
- TRANSFER BOTH, SEC1,SEC2
- TRANSFER ALL,EJE1,EJE3,4
- TRANSFER PICK,PRIMERO,ULTIMO
- TRANSFER FN,LUGAR,3
- TRANSFER P,LUGAR,2
- TRANSFER SBR,REG,MARC
- TRANSFER SIM,NORET,RET

TRANSFER ALL,EJE1,EJE3,4

```
EJE1    SEIZE MAQ01
        ADVANCE 10
        ADVANCE 5
        RELEASE MAQ01

EJE2    SEIZE MAQ02
        ADVANCE 12
        ADVANCE 0
        RELEASE MAQ02

EJE3    SEIZE MAQ03
        ADVANCE 12
```

Example: TalsaV1.0

- Two automatic lathes.
- Arrivals (4 ± 1 uniform).
- Lathe A: 1 to 10 minutes (uniform).
- Lathe B: 2 to 15 minutes (uniform).
- Pieces enters in the first free, (we prefer the A).
- Simulate 50 pieces.

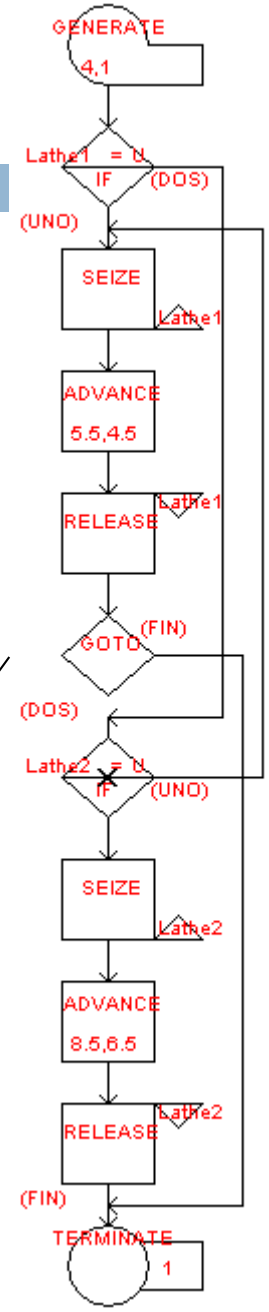
Example: TalsaV1.0 (answer)

```
                SIMULATE

                GENERATE    4,1
                QUEUE       MATERIAL
                TRANSFER    BOTH,UNO,DOS
UNO             SEIZE       TALAD1
                DEPART      MATERIAL
                ADVANCE     5.5,4.5
                RELEASE     TALAD1
                TRANSFER    ,PROD
DOS             SEIZE       TALAD2
                DEPART      MATERIAL
                ADVANCE     8.5,6.5
                RELEASE     TALAD2
PROD            TERMINATE   1

                START       50
                END
```

```
LABEL

UNO

DOS

FIN
```

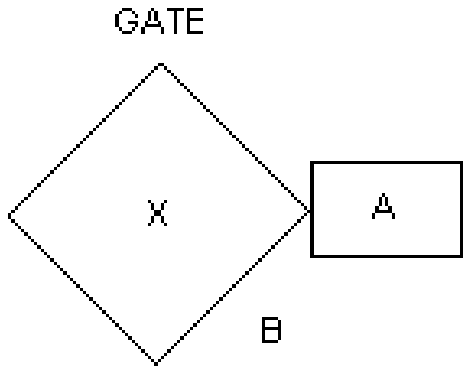
LOC	BLOCK TYPE	ENTRY COUNT
1	GENERATE	51
2	QUEUE	51
3	TRANSFER	51
4	SEIZE	35
5	DEPART	35
6	ADVANCE	35
7	RELEASE	35
8	TRANSFER	35
9	SEIZE	16
10	DEPART	16
11	ADVANCE	16
12	RELEASE	15
13	TERMINATE	50

```
Clock        208.51

Block counts
Number Adr.   Oper.  Current  Total
    1         GENERA           53
    2         IF               53
    3  UNO    SEIZE            33
    4         ADVANC           33
    5         RELEAS           33
    6         GOTO             33
    7  DOS    IF         2     47
    8         SEIZE            18
    9         ADVANC     1     18
   10         RELEAS           17
   11  FIN    TERMIN           50
```



IF with station (Gate on [W])

- Controls the transaction flow.
- A: name or number of the analyzed installation.
- B: name of the label.
- X: Auxiliary operator.
- GATE NU INST,ALT



Gate [W] (2/2)

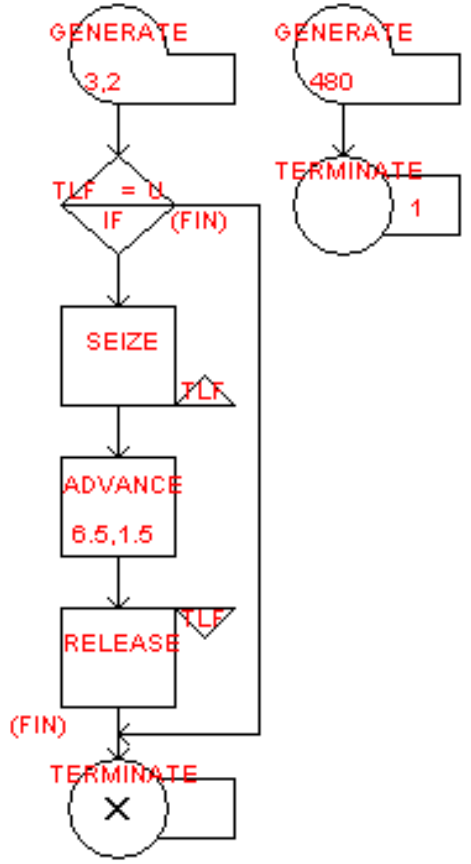
- Related to SEIZE i RELEASE
  - U Try if the installation is full.
  - NU Try if the installation is free.
- Related to ENTER i LEAVE
  - SF: Try if the server is full.
  - SNF: Try if the server is not full.
  - SE: Try if the server is empty.
  - SNE: Try if the server is not empty.
- Related to LOGIC
  - LS: Set logic
  - LR: Reset logic.

Example: ViatgesV1.0

- The clients call the travel agency following a uniform distribution (3 ± 2 minutes).
- Give the information to the clients follows a uniform distribution of 5 to 8 minutes.
- If the telephone is occupied the client is lost.
- Simulate 8 hours.

Example: ViatgesV1.0 (answer)

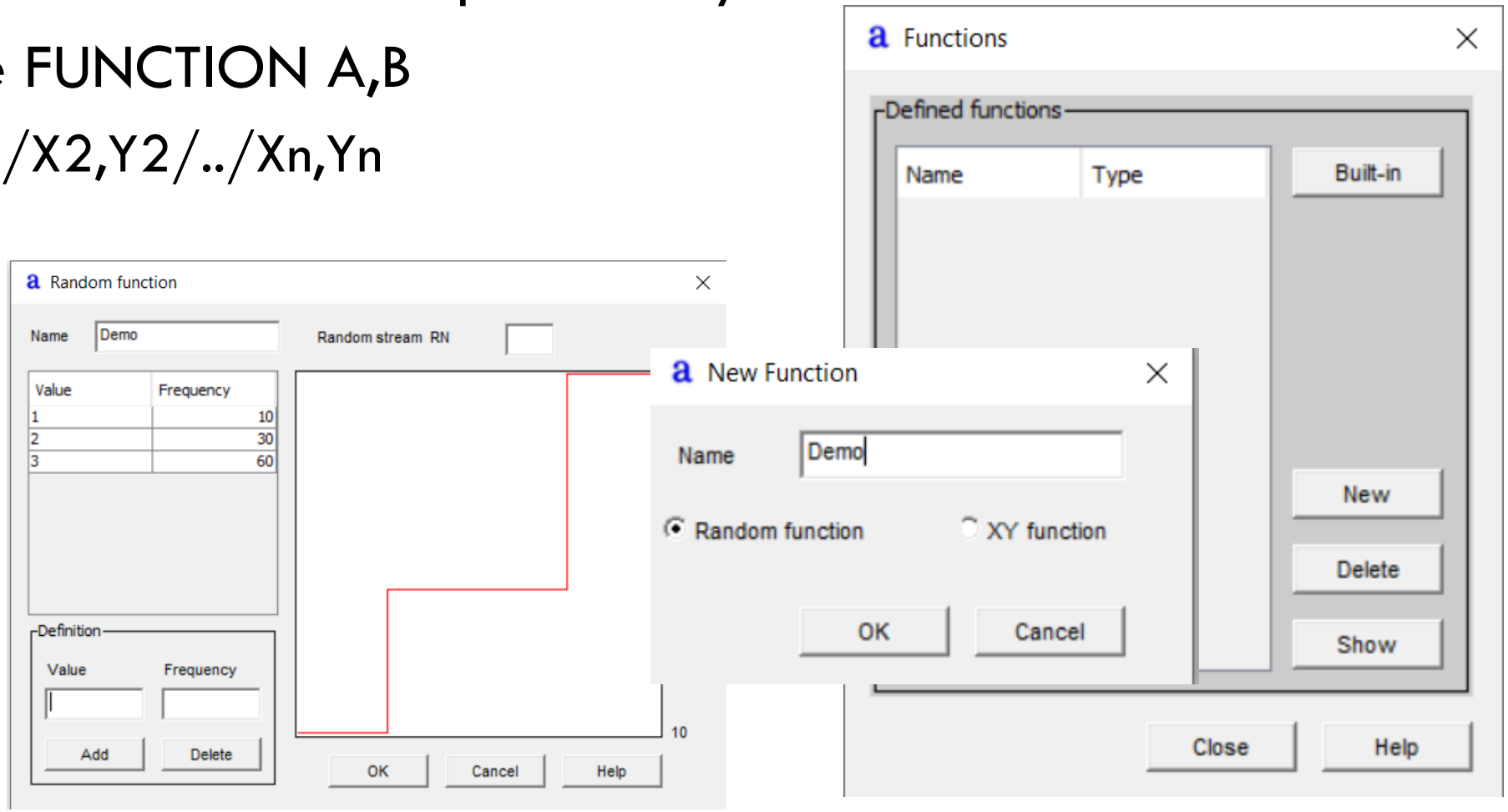
```
        SIMULATE
                GENERATE        3,2
                GATE NU         TELEF,NEXT
                SEIZE           TELEF
                ADVANCE         6.5,1.5
                RELEASE         TELEF
NEXT            TERMINATE
                GENERATE        480
                TERMINATE       1
                START           1
```



FUNCTION

- Allows to define a new probability distribution.
- Name FUNCTION A,B

  X1,Y1/X2,Y2/../Xn,Yn



FUNCTION

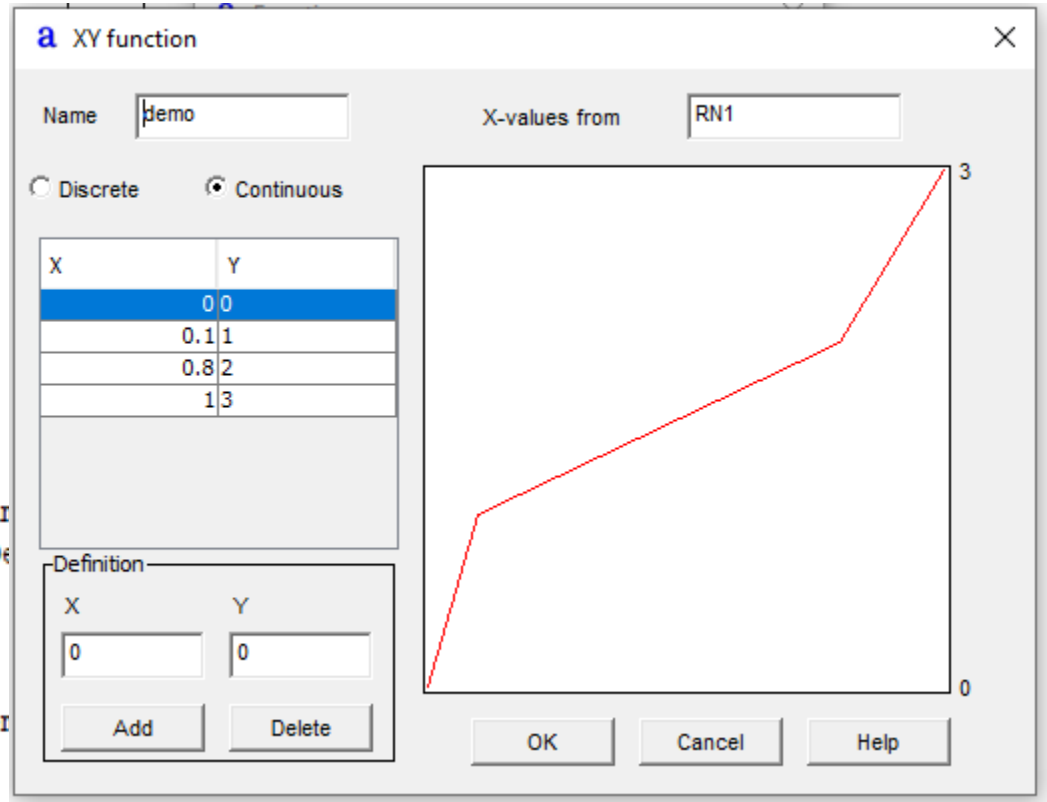
- Nom: Reference name of the function.
- A: Function arguments.
- B: Type of the function.
  - (C,D,E,L,M).
- Xi,Yi: Pair of data to create the distribution function.
  - Xi reference value.
  - Yi is the value that the function returns.

FUNCTION C

- Continuous.
  - Given an X value, interpolates and returns a value for Y.
  - As an example:
    - A=RN1
    - The function must be defined between 0 and 1
    - MyFuncName FUNCTION RN1,C3
    - 0.1,1/0.8,2/1,3

```
demo FUNCTION RN1,C
0 0
0.1 1
0.8 2
1 3
```

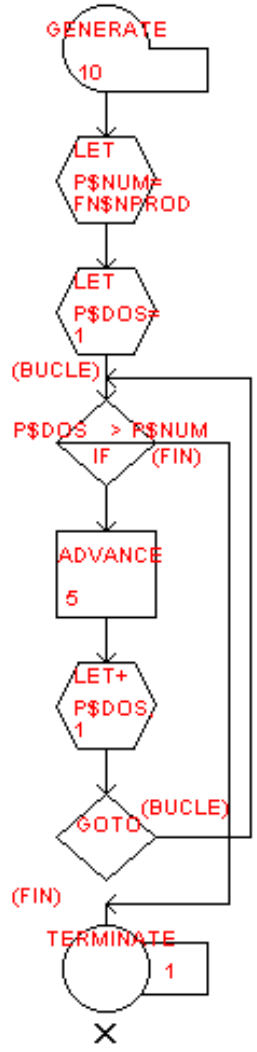
```
TIEMPO_LLEGADA FUNCTION RN1,C5          ; Argur
    0,5 / 0.25,7 / 0.5,9 / 0.75,11 / 1,13 ; De
                                GENERATE FN$demo
          SIMULATE              TERMINATE 1

          GENERATE    FN$TIEMPO_LLEGADA    ; Tier
          TERMINATE   1
                      START 100
                      END
```



FUNCTION D

- Discrete.
- Growing values of X.
- If we find a value equals or greater than X we return its related
- If we do not find this value, returns the greater value.

```
NPROD FUNCTION RN1,D
0.3 1
0.6 2
0.8 3
1 4


          GENERATE 10           ; Genera una transacción
          LET P$UNO=FN$NPROD       ; Asigna el número de productos basado en la función discreta
          LET P$DOS=1             ; Inicializa un contador para el bucle (P$DOS)
BUCLE     IF P$DOS>P$UNO,FIN  ; Si P$DOS es mayor a P$UNO, ir a FIN (equivalente a repetir todo el bloque P$UNO ve
          ADVANCE 5                ; Simula el procesamiento de un producto
          LET+ P$DOS,1             ; Incrementa el contador P$DOS
          GOTO BUCLE              ; Vuelve al inicio del bucle
FIN       TERMINATE 1

START 50
END
```



FUNCTION D (W)

- Discrete.
- Growing values of X.
- If we find a value equals or greater than X we return its related value.
- If we do not find this value, returns the greater value.

```
N_PROD FUNCTION  RN1,D4                ; Argumento: RN1 (0-1), Discreta con 4 puntos
    0.3,1 / 0.6,2 / 0.8,3 / 1,4        ; Definición de la función

        SIMULATE

          GENERATE  10                 ; Genera una transacción
          ASSIGN    1,FN$N_PROD        ; Asigna el número de productos basado en la función discreta
          ASSIGN    2,1                ; Inicializa un contador para el bucle (P$2)
BUCLE     TEST LE   P2,P1,FIN          ; Si P2 es mayor a P1, ir a FIN (equivalente a repetir todo el bloque P1 veces)
          ADVANCE   5                  ; Simula el procesamiento de un producto
          ASSIGN    2+,1               ; Incrementa el contador P$2
          TRANSFER  ,BUCLE             ; Vuelve al inicio del bucle
FIN       TERMINATE 1

          START     50
```

Si se usan etiquetas con palabras
ASSIGN DOS,1
TEST LE PDOS, PUNO, FIN

FUNCTION E [W]

- Discrete function of attribute value.
  - Returns for an X the attribute value.
  - RESUL FUNCTION X$VALOR,E3
    1,S$ALM1/5,S$ALM2/9,S$ALM3

FUNCTION L [W]

- Value list
- Returns the value of the X position (argument)
- TIPUS FUNCTION P2,L4

  1,3/2,5/3,8/4,12

FUNCTION M [W]

- Attribute value list
- Returns the value of the attribute in the position X (argument)
- LLISTA FUNCTION X$NOM,M3

  1,X$NOM1/2,X$NOM2/3,X$NOM3

Functions main aspects

1. Functions C,D,L do not admit SNA's and Y's.
2. Functions E, M must have SNA's as Y values.
3. Functions L and M cannot use random arguments.
4. To use a function:
    1. **FN$nom**
    2. FN(nom).
    3. F$nom(parametres).

Example: Wooden tool v1.0

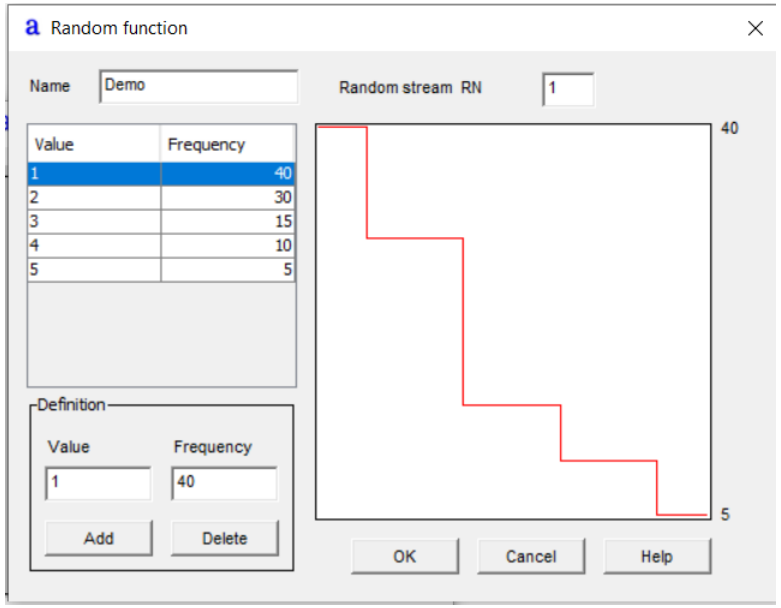
- Arrivals 5 a 9 minutes (Uniform)
- Tool service time (minutes)

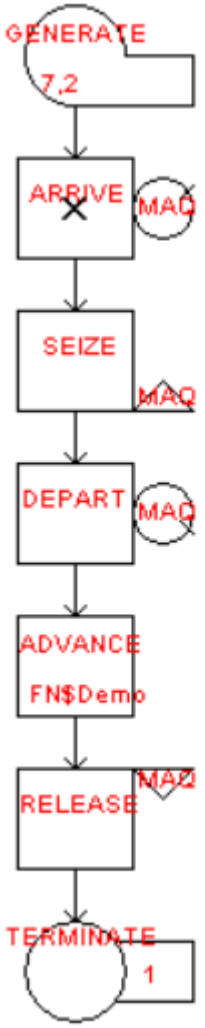
Time	1	2	3	4	5
Probability	.4	.3	.15	.10	.05

- Model this system during 8 hours.

Resposta Serreria V1.0

```
        SIMULATE
TRAB        FUNCTION   RN1,D5
.4,1/.7,2/.85,3/.95,4/1,5
        GENERATE        7,2
        QUEUE           UNO
        SEIZE           MAQ
        DEPART          UNO
        ADVANCE         FN$TRAB
        RELEASE         MAQ
        TERMINATE
*
*Termination control blocks
*
        GENERATE        480
        TERMINATE       1
        START           1
```





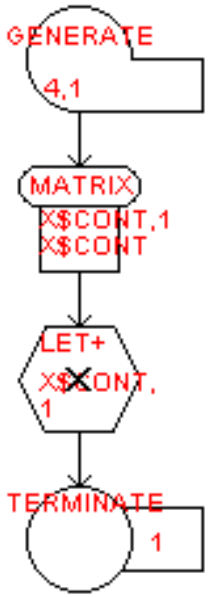
Start Values

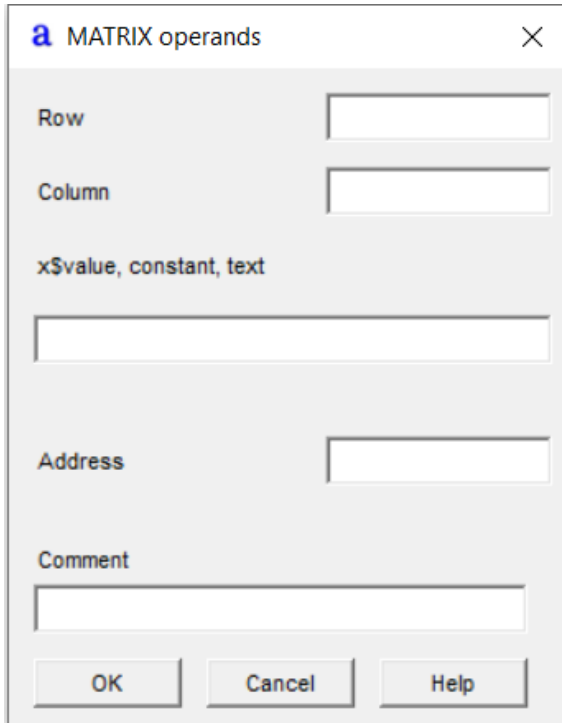
- “Control” -> “Start values”
- X$nom
- Establish initial simulation conditions
- Some variables need to be defined from the beginning

Matrix

- Name MATRIX A,B,C
- A: Matrix type.
- B: Rows.
- C: Columns.
  - MAGATZEM MATRIX MH,200,4
  - Defines a 200 x 4 matrix.

```
LET X$CONT=1
        GENERATE 4,1
        MATRIX X$CONT,1,X$CONT
        LET+ X$CONT,1
        TERMINATE 1

START 100
END
```





Msavevalue [W]

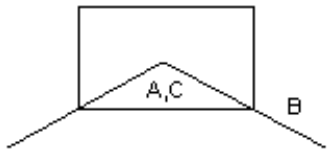
- To give or modify the value of a matrix.
- A: name.
- B: row number.
- C: column number.
- D: information to be stored.

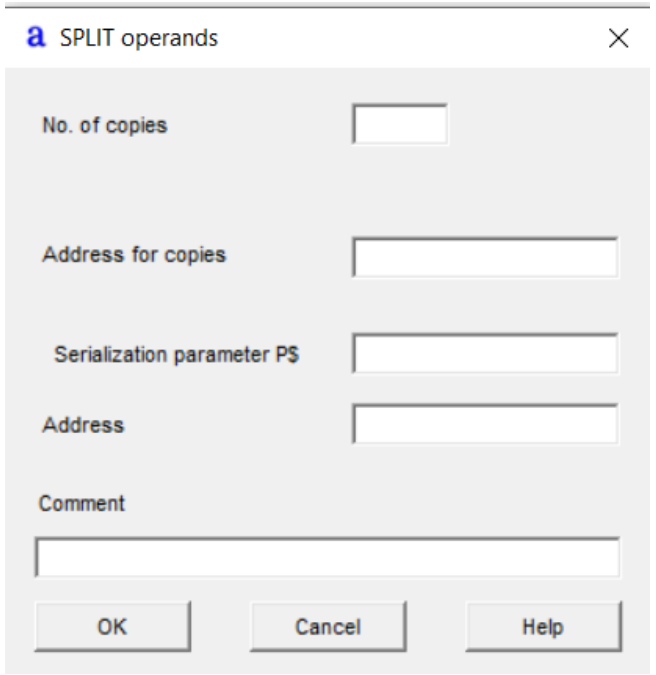
Split

- Allows the creation of new transactions with the same features of active transaction.
- A: N° of new created transactions.
- B: Destination of the new transactions (op).
- C: Parameter that receives the serial number.

- SPLIT 10,COPYs,SERIAL
- ADVANCE 5 ;1 XACTS
- TRANSFER, ENDSIM

- COPYs ADVANCE 10 ;10 XACT





Example: TaladreSplit V1.0

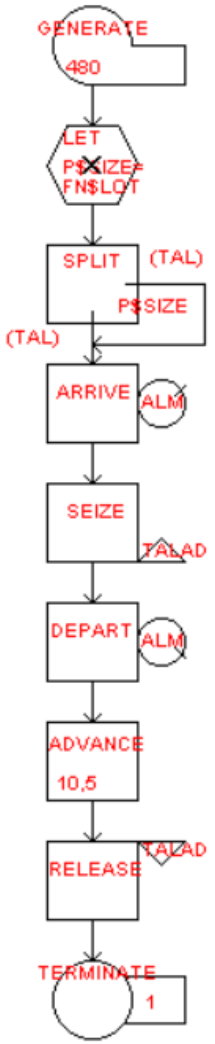
- Entities every 8 hours.
- Size of the lotes:

Lot size	17	18	19	20	21
Probability	0.1	0.4	0.4	0.05	0.05

- Service time 10 ± 5 (in minutes).
- Simulate 3000 pieces

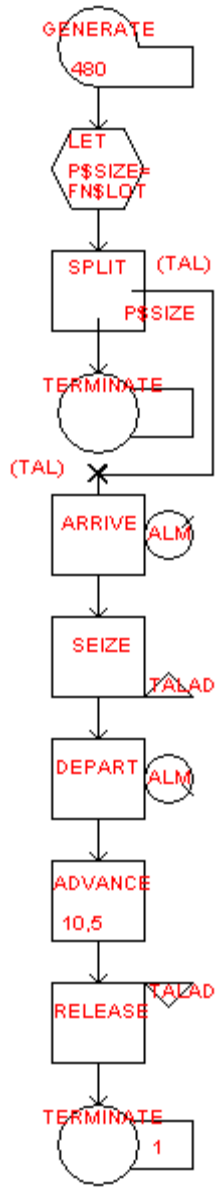
Example: TaladreSplit V1.0 (sample)

```
LOT FUNCTION            RN1,D5
.1,16/.5,17/.9,18/.95,19/1,20

        SIMULATE
*
*       ONE-LINE, SINGLE-SERVER QUEUEING MODEL
*
        GENERATE        480
        SPLIT           FN$LOT,TAL
TAL     QUEUE           ALM
        SEIZE           TALAD
        DEPART          ALM
        ADVANCE         10,5
        RELEASE         TALAD
        TERMINATE       1
*
        START           3000
        END
```



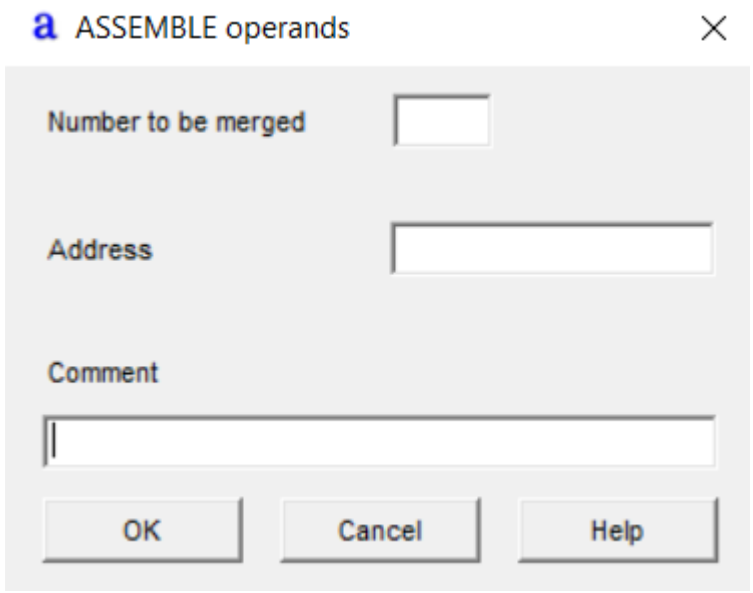
Example: TaladreSplit V2.0 (sample)

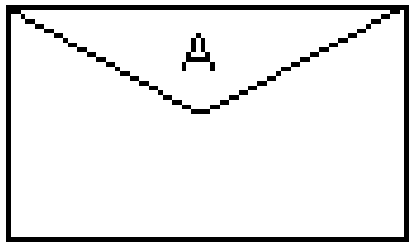
```
LOT FUNCTION            RN1,D5
.1,17/.5,18/.9,19/.95,20/1,21

        SIMULATE
*
*       ONE-LINE, SINGLE-SERVER QUEUEING MODEL
*
        GENERATE        480
        SPLIT           FN$LOT,TAL
        TERMINATE
TAL     QUEUE           ALM
        SEIZE           TALAD
        DEPART          ALM
        ADVANCE         10,5
        RELEASE         TALAD
        TERMINATE       1
*
        START           3000
        END
```

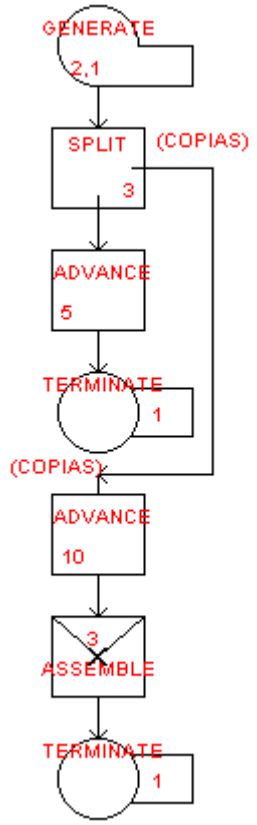


Assemble

- To synchronize transactions.
- A: Number of transactions we are looking for.

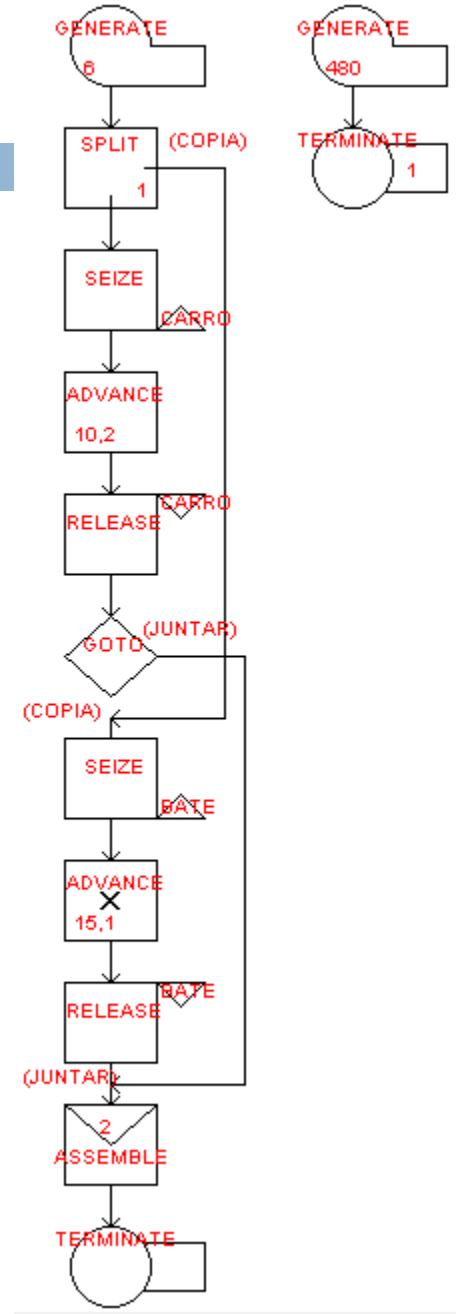






Exercise: CarFactory v1.0

- En una fábrica de coches, fabricamos por separado y en paralelo la carrocería y la batería cuyas partes supondremos que nos llegan simultáneamente. Cuando ambos elementos estén listos estas se deben unir para completar el vehículo. Simula el proceso teniendo en cuenta que la carrocería ocupa un robot que tarda de 8 a 12 minutos en completarla y la batería ocupa otro distinto que tarda de 14 a 16 minutos. Las transacciones se generan a intervalos regulares de 6 minutos.
- Considerad una jornada de 8h (480m) de simulación.



Internal vision of the transaction's movement

Understanding the process interaction paradigm

Example (Blocs)

1. Entering 3 ± 1 minutes
2. Start Storage
3. Entering Resource
4. Exit Storage
5. Using the resource 3
6. Release resource
7. Exit system

Example (Programming blocs)

1. New entities arrivals
   - 3 ± 1 minutes
2. Verification and *capture* of the free resource
3. Using the resource
   - 3 minutes
4. Release the resource
5. The entity leaves the system

Example(+ statistical adquisition)

1. New entity arrival
   - 3 ± 1 minutes
2. Start of the acquisition of data to represent the accumulation
3. Verification and capture of the free lathe
4. End of the data acquisition to represent the accumulation
5. Turning the raw material
   - 3 minutes
6. Release the lathe
7. Exit the system

Example (Event chains)

1. Enters a new transaction on the system
   - On t enters the new entity i+1 to the future event chain, remaining here until t+u(2,4).
2. Verification of the lathe entrance
   1. If the entity enter the lathe continues its movement to the next block
   2. If the lathe is not free, the entity is send to the end of the current event chain, remaining here until the lathe be free
3. Entering in the future event chain, remaining here 3 minutes
4. Leaving the future event chain, the lathe is free
5. The entity leaves the system

Points of view of a GPSS model

- External vision of the transactions. From the point of view of the block programming
  - The set of blocks that defines the movement of the transactions
- Internal vision of the transactions. From the point of view of the event chains
  - The places where the transactions are send during its movement through the model.

Event chains

- Transactions list
- In any moment
  - Transaction $\in$ bloc
  - Transaction$\in$ chain
- The transaction makes it movement from:
  - One block to another: no blocking situation, no delay.
  - From a chain to a chain: blocking situation, usually form FEC to CEC
  - From a block to a chain: A blocking situation or a delay in the system (ADVANCE)
  - From a chain to a block: An unblocking situation (or the end of a delay)

Blocking in the event list

- Blocking due to a delay
  - The transaction enters in the block in t1 and leaves the block in t2 (typically an advance)
  - In GPSS only due to ADVANCE and GENERATE.
- Blocking due to a model condition
  - The resource is “full”, typically a SEIZE used by any other entity

Type of chains

1. Current events chain
2. Future events chain

Current event Chain (CEC)

- Contains the transaction that want move now
  - Some problems prevents this movement
    - Blocking situations
    - Server busy
  - Sorted by decreasing priority (no time)

CEC

- Move time: Current simulation time

xact id	curBlk	nxtBlk	moveTime	priorityLevel
5	7	8	...	20
3	12	13	...	16
8	9	10	...	12

Future event Chain (FEC)

- The transactions are waiting for the correct time to finish its actions
- Can be caused by
  - A new transaction enters in the model, GENERATE
  - The transaction is in a process delay, ADVANCE
- Sorted by time and priority

FEC

- 7,2,11 : blocks ADVANCE
- 9 : block GENERATE

xact id	curBlk	nxtBlk	moveTime	priorityLevel
7	3	4	42.6	3
9	Neix	19	47.6	15
2	7	8	51.9	12
11	32	33	51.9	16

Example GPSS

GPSS World Simulation Report - TaladreSplit V1.0.3.1

Tuesday, March 08, 2005 10:40:14

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	493.810	8	1	0

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1	0	0
	2	SPLIT	1	0	0
TAL	3	QUEUE	18	16	0
	4	SEIZE	2	1	0
	5	DEPART	1	0	0
	6	ADVANCE	1	0	0
	7	RELEASE	1	0	0
	8	TERMINATE	1	0	0

Example GPSS

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
TALAD	2	0.028	6.905	1	3	0	0	0	16

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
ALM	17	17	18	1	0.475	13.043	13.810	0

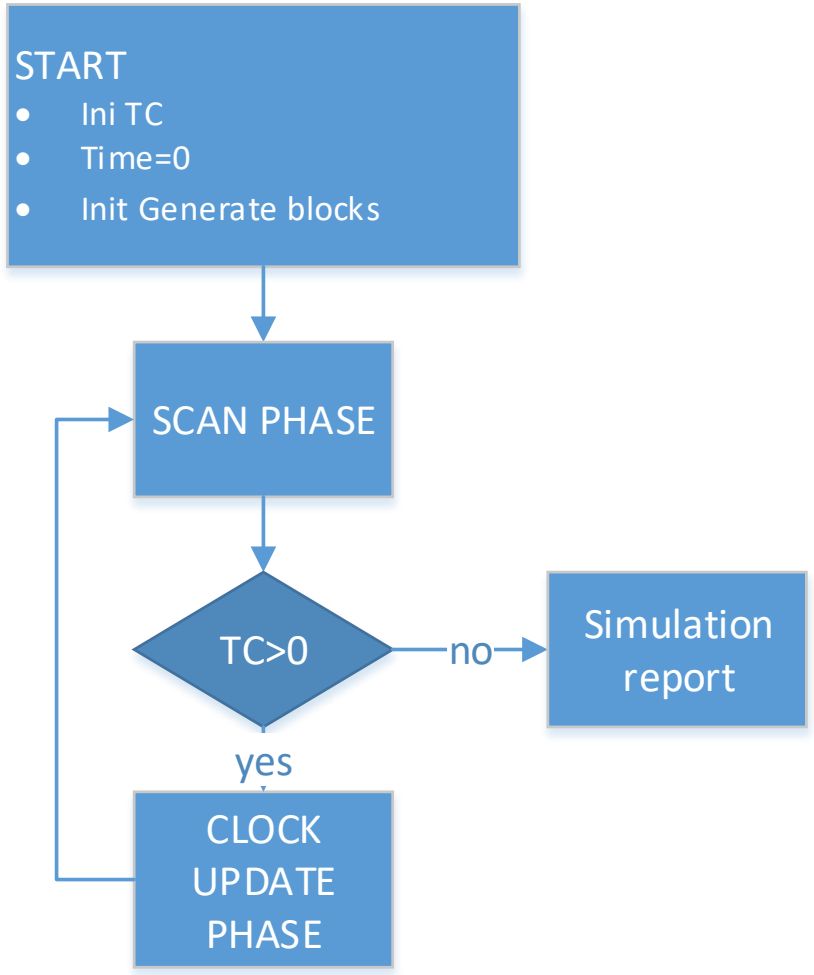
CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
3	0	480.000	1	4	5		

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
2	0	960.000	2	0	1		

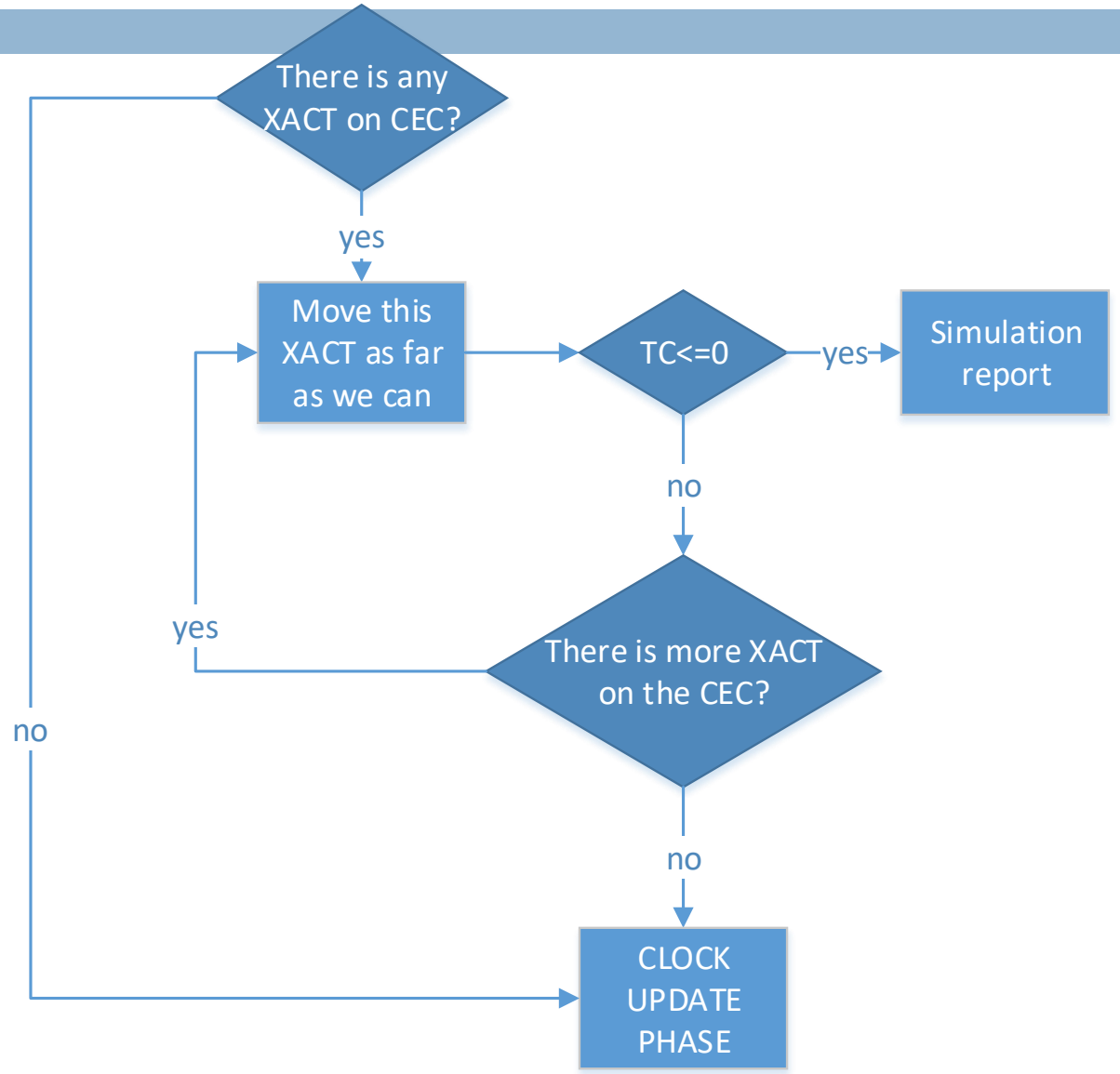
GENERATE blocs initialization

- On time 0.
- In Top-Down order (GPSS/H)
- For each bloc one transaction are created.
- Identifiers are assigned consecutively.
- Assigning the moveTime for each transaction.
- If the moveTime is equals to 0, this transaction I queued in the CEC, otherwise in the FEC.

Transactions movement

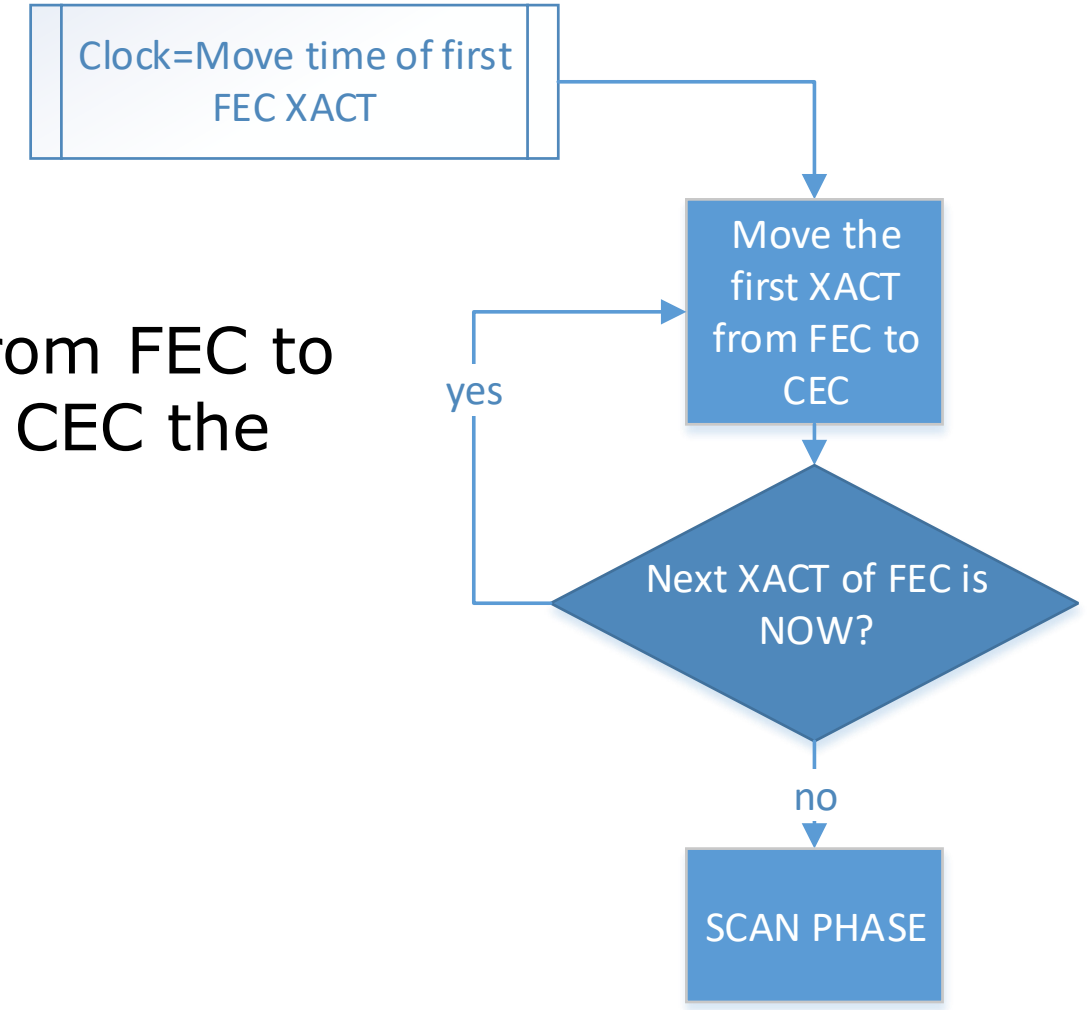


SCAN PHASE


There is any XACT on CEC?
yes
Move this XACT as far as we can
TC<=0
yes
Simulation report
no
There is more XACT on the CEC?
yes
no
no
CLOCK UPDATE PHASE

UPDATE PHASE

Moving the XACT from FEC to CEC keeping in the CEC the priority order.



Example (Blocks)

1. Enter 3 ± 1 minutes
2. Start Store
3. Entering Lathe
4. Leaving Store
5. Turning 3
6. Exit Lathe
7. Exit System

Example (data)

- Interval between generations:
  - (2,2,4,4)
- We only generate 4 entities.

1.	Enter	3 ± 1 minutes
2.	Start	Store
3.	Entering	Lathe
4.	Leaving	Store
5.	Turning	3
6.	Exit	Lathe
7.	Exit	System

Steep	Time	CEC	FEC
1	Start	-	-
2	0	-	(1,Out,1,2)

Example (event chains)

Step	Time	CEC	FEC	Comments
1	Inici	-	-	
2	0	-	(1,Out,1,2)	First Xact.
3	2	(1,Out,1,Now)	-	Xact from FEC to CEC.
4	2	-	(2,Out,1,4) (1,5,6,5)	Moving the Xact 1 all that we can, entering in 5 (*advance)*. Generatio of the second Xact.
5	4	(2,Out,1,Now)	(1,5,6,5)	Xact from FEC to CEC.
6	4	(2,2,3,Now)	(1,5,6,5) (3,Out,1,8)	Moving the Xact 2 all that we can, entering the 2 (*seize). Generation of the third* Xact.

Example (event chains)

Step	Time	CEC	FEC	Comments
7	5	(2,2,3, now) (1,5,6, now)	(3,Out,1,8)	Xact from FEC to CEC.
8	5	-	(3,Out,1,8) (2,5,6,8)	Moving the Xact 1 all that we can, leaving the system. Moving the Xact 2 all that we can, entering the 5 (*advance*).
9	8	(3, Out,1,now) (2,5,6, now)	-	Xact from FEC to CEC.
10	8	-	(3,5,6,11) (4,Out,1,12)	Moving the Xact 2 all that we can, leaving the system. Moving the Xact 3 all that we can, entering the *5(advance*). Programming the next arrival.

Example (event chains)

Step	Time	CEC	FEC	Coments
11	11	(3,5,6,Now)	(4,Out,1,12)	Xact from FEC a CEC.
12	11	-	(4,Out,1,12)	Moving the Xact 3 all than we can, leaves the system.
13	12	(4,Out,1,Now)	-	Xact from FEC a CEC.
14	12	-	(4,5,6,15)	Moving the Xact 4 all that we can, entering the 5 bloc (*advance*).
15	15	(4,5,6,Now)	-	Xact from FEC to CEC.
16	15	-	-	Moving the Xact 4 all that we can, leave the system.