GPSS

Pau Fonseca i Casas; pau@fib.upc.edu

GPSS

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

GPSS world

- Entities (transactions) traveling through the system.
- Through the blocs.
  - The number of blocs is different depending on the GPSS version used.

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.

Files:

- GPSS/H version:
  - .gps (containing the model)
  - .lis (containing the results of the model execution)

Language structure

- 4 kind of instructions
  1. System access instructions
  2. Variable definition instructions
  3. Program logic instructions
  4. Simulation control instructions

System access instructions

- GPSSH [file.gps] TV.
  - To obtain the simulation control.
  - Display.
  - Trap: breakpoints.
  - Set:
    - TV off→ All the screen for the dialog window.
    - TV on→ Shows the 3 windows.

Display

- PF→ Function keys.
- Blo→ Actual and total blocs.
- CEC→ Current event chain.
- Clocks→Absolut and relative clock.
- FEC→ Future event chain.
- Xact=“id”→ Features of the current transaction.

Trap

- Trap Scan→Breakpoint in the start of the Scan Phase.
- Untrap Scan→ To delete the breakpoint.

Variable definition instructions

- Functions definition (FUNCTION)
- Machine number definition (STORAGE)
- Matrix definition (MATRIX)
- Numerical assignation of variables (EQU)
- Variable initialization (INITIAL)
- Histogram definition (TABLE)
- Operations definition (VARIABLE i FVARIABLE)

Program logic instructions

- Named blocs.

Simulation control instructions

- START
- END
- SIMULATE

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
    END
```

Blocs (I)

- Permanent and static entities (do not flow through the model).
- Used by transactions to do some jobs.
  - Facilities (1).
  - Storages (n).

Blocs (II)

- Describing how the entity flows throw the model.
- Representing action or event.
- Combination of blocs → process defining what happens to a transaction → model logic.
- Graphical representation.
  - Clear explanation.
  - Helps in the design.

Entity (Transaction on GPSS)

- Destination route.
- Related statistics.
  - Blocs visited.
  - Waiting time.
- Kind.

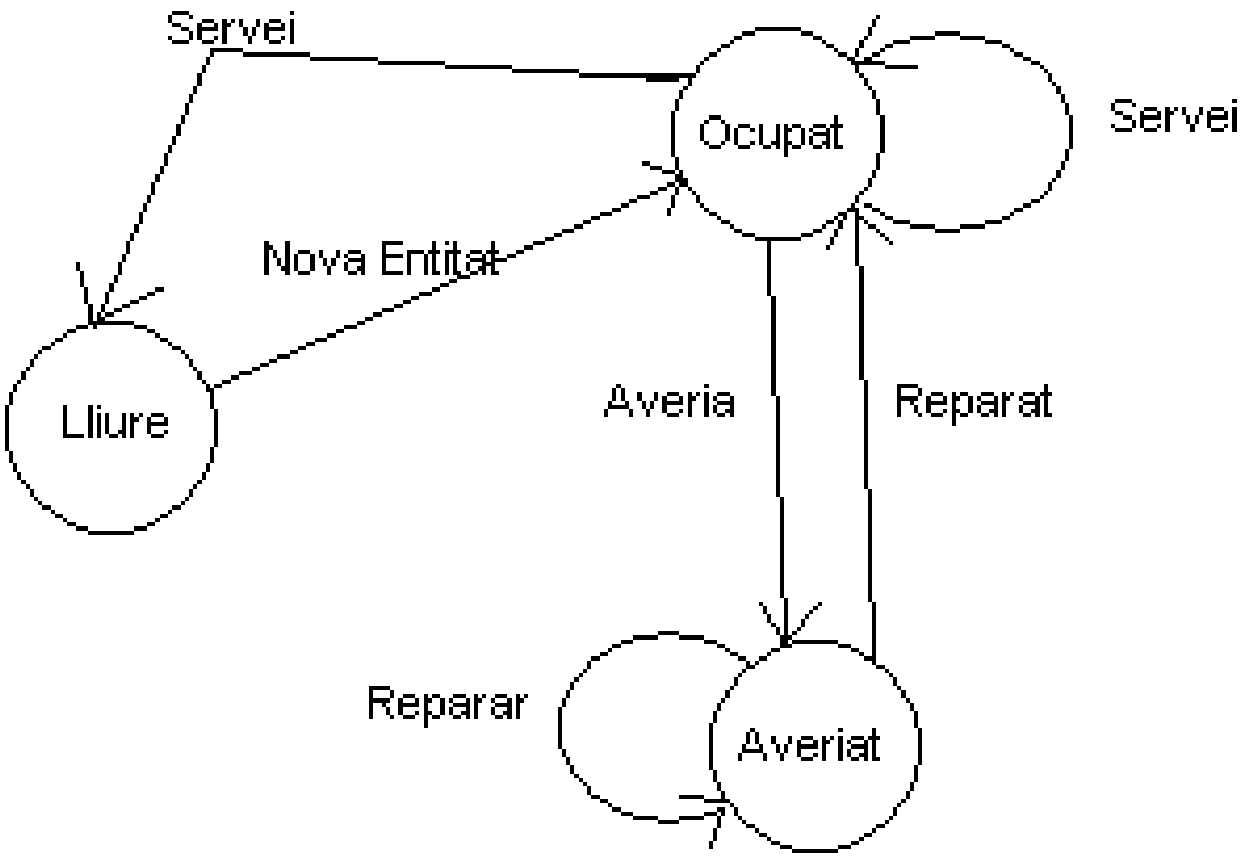
Simulation object

- State
- Number of elements in the queue.
- Related statistics.
- Kind of object.

Event

- Creation time.
- Execution time.
- Priority
- Kind of event.
  - Depending on the kind of event a simulation element develops one action or other.

Modification in the state of a simulation element.



Blocs

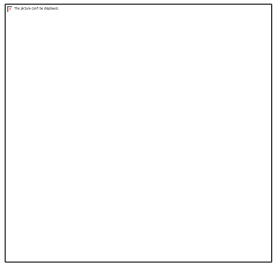
Program logic instructions

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
- F: Number of parameters.

Terminate

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

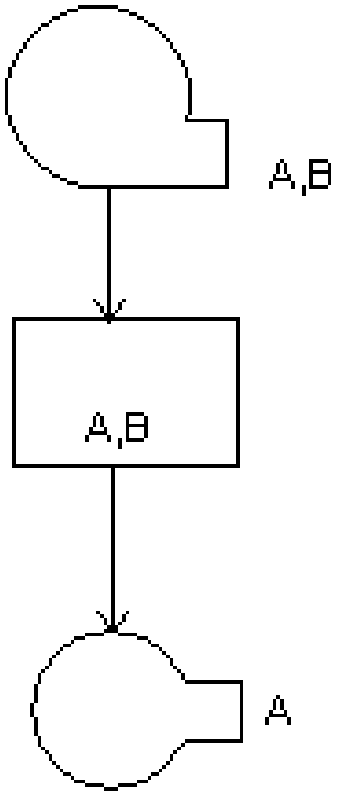


Advance

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

Example

- Museum

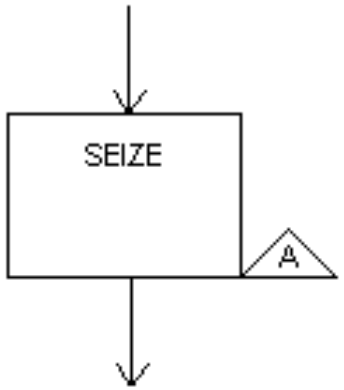


Modeling simple servers

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

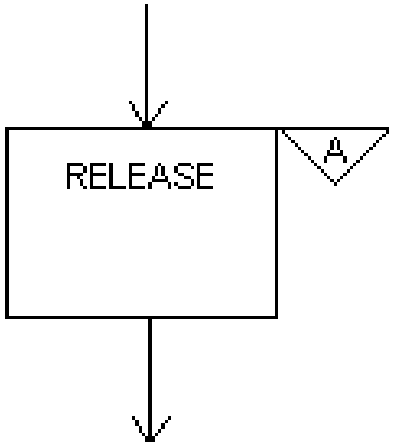
Seize

- 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)

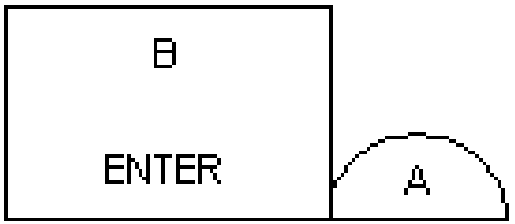
- GENERATE 7,3
- SEIZE TORN
- ADVANCE 5,2
- RELEASE TORN
- TERMINATE 1

Modeling complex servers

- Is needed to define the server capacity.
- STORAGE S(ELEVATOR),6
- ELEVATOR STORAGE 6
- Is needed to show when the server is requested and when the server is released.

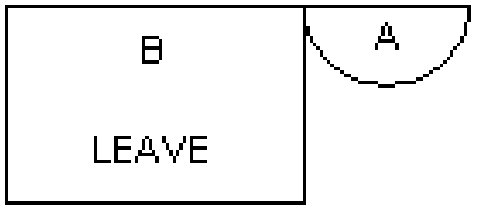
Enter

- 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.



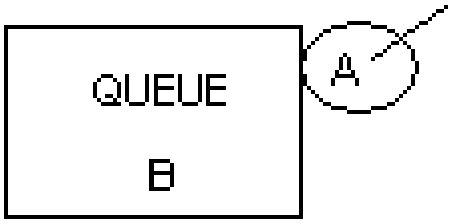
Leave

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



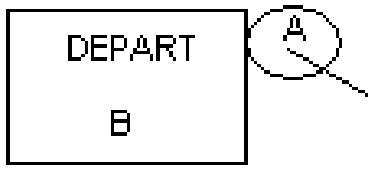
Queue

- To model the queues in front of a server.
  - A: queue identifier.
  - B: number of elements entering in the queue. Optional, 1 by default.



Depart

- To show that an entity is leaving a queue.
  - A: queue identifier.
  - B: number of elements leaving the queue. Optional, 1 by default.



Queue Reports (I)

- Queue: queue identifier.
- Max Count: queue maximum contents.
- Avg count: queue average contents.
- Total entries: queue total entries.

Queue Reports (II)

- Zero entries: entries with delay time = 0.
- Percent Zeros: % of entries that are *zero entries*.
- Avg Time: average time of stay in the queue.
- $Avg Time: average time without the *zero entries*.

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.

Example: Banc Fortuna v1.0 (answer)

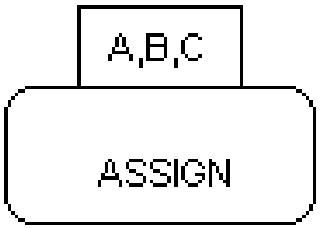
- GENERATE 7,2
- QUEUE CUA
- SEIZE CAIXER
- DEPART CUA
- ADVANCE 4,2
- RELEASE CAIXER
- TERMINATE 1

Example: Banc Fortuna v1.1 (answer)

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

Assign

- Allows the modification of the transaction parameters.
- A: parameter's number.
- B: value to assign.
- C: kind of the parameter.
  1. PH→half word.
  2. PF→full word.
  3. PL→floating point.
  4. PB→byte.



Labels

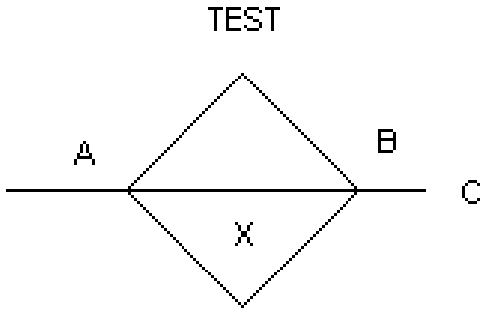
- Is allowed to name the GPSS blocs.
  - To access the SNA's.
  - To break the transaction sequence.

SNA's

- Some information related to the model entities.
- Can be used in simulation time.
- Give information about the simulated model.
- Examples:
  - C1: Clock
  - N$label : #Xacts

Test

- 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

- 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: bigger
  - GE: bigger or equal.
  - L: les
  - LE: les or equal.
  - NE: no equal.

Example: Banc Fortuna V3.0

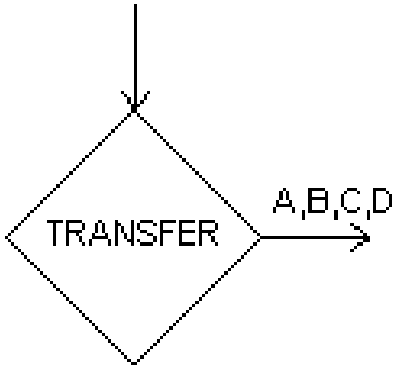
- In a banc the clients arrive following an uniform distribution with parameters 5 to 10 (minutes).
- 3 tellers.
- Service time: 2 to 5 minutes (uniform distribution).
- Simulate 1 day of work.
- At the end of the day no client must remain in the banc.

Example: Banc Fortuna V3.0 (answer)

```
                SIMULATE
                STORAGE     S(CAIXES),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

*
*Blocs de control de terminació
*
        GENERATE            240
        TEST E              N(ENT),N(SORT)
        TERMINATE           1
        START               1
        END
```

Transfer

- Allows to break the sequential movement of a transaction.



Transfer

- A:tranference modality
  - Both, All, Pick, FN, P, SBR, SIM, Fraction, Number, SNA, Null
- Optional parameter.

Transfer

- B: number or bloc position.
- C: number or bloc position.
- D: number or bloc position.

Transfer

- TRANSFER .40,OPC1,OPC2
- 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

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
```

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.

FUNCTION D

- 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.

FUNCTION E

- 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

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

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

FUNCTION M

- 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 important aspects

1. Functions C,D,L do not admit SNA's ans 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. F$nom(parametres).

Example: Wooden tool v1.0

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

Temps de procés	1	2	3	4	5
Freqüència relativa	.4	.3	.15	.10	.05

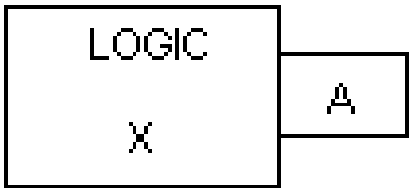
- Model this system during 8 hours.

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
      END
```

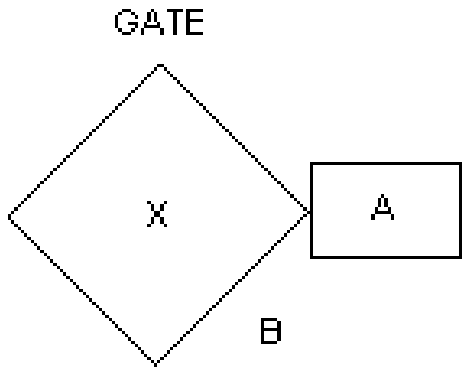
Logic

- Allows the modification of the logic bloc that represents.
  - X: Logic operator.
    - S (set)
    - R (reset) By default
    - I (Invert)
  - A: logic control identifier.



Gate (1/2)

- 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 (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

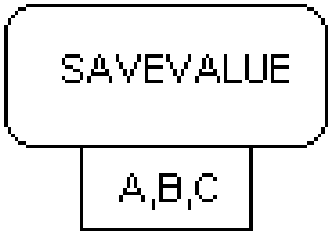
- The clients call the travel agency following an uniform distribution (3 ± 2 minutes).
- Give the information to the clients follows an uniform distribution of 5 to 8 minutes.
- If the telephone is occupied the client is lost.
- Simulate 8 hours.

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
```

Savevalue

- To give or modify the value of a SAVEVALUE element.
- A: SAVEVALUE name.
- B: Value assigned to the SAVEVALUE (integer, name or SNA).
- C: SAVEVALUE type:
  - XH→ half word.
  - XF→ full word.
  - XL→ floating point.
  - XB→ byte.



Accessing to a SAVEVALUE

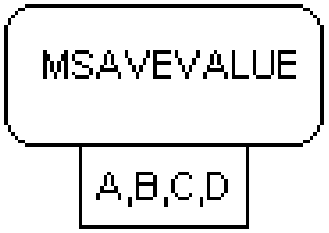
- We can access the value stored in a SAVEVALUE in any part of the GPSS program through the sentence:
  - X(nom) (XH, XF, XL, XB) [H]
  - X$nom [W]

Matrix

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

Msavevalue

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



Initial

- To initialize the LOGICSWITCH, SAVEVALUE or the matrix.
- INITIAL LSMyLogic,1.
- INITIAL XH(MySavevalue),10.
- INITIAL XF(1),10.
- INITIAL XL(1),10.
- INITIAL XB(1),10.
- INITIAL MX$nom(1,2),5.

Ampervariables (definition)

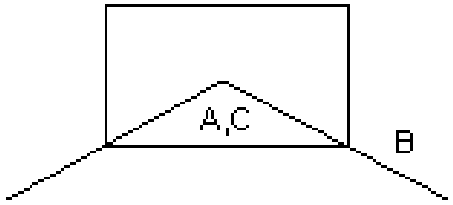
- Global variables
- INTEGER &I,&D(10)
- REAL &Pes
- CHAR*10 &C
- VCHAR*4 &Títol

Ampervariables (Initialization)

- LET.
  - LET &var=A.
- GETLIST.
  - GETLIST &var1,&var2.
    - The user interacts with the model adding some information.
- Can be used as a blocs.
  - BLET.
  - BGETLIST.

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.



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

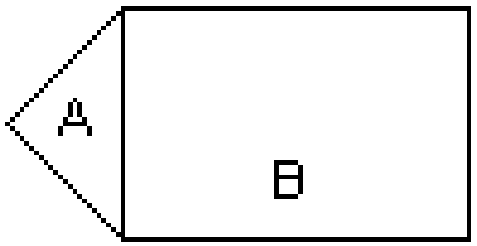
- Service time 10 ± 5 (in minutes).
- Simulate 3000 pieces

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
```

Loop

- Allows to modify the destination of an active transaction.
- A: Parameter containing the number of times a transaction passes an specific section.
- B: Destination.



Example: Loop

```
                ASSIGN      voltes,10
REG             ENTER       SERV
                ..
                LOOP voltes,REG
                SEIZE  CAJERO
```

Example: Wagons V1.0

- 5 transport wagons (of pieces) between two points.
- Initial point: loaded by 1 worker with 50 pieces. U(5,7) seconds x piece.
- Movement to the final point U(4,8) minutes.
- Download by a second worker. U(10,16) seconds x piece.
- Movement to the origin U(3,7) minutes.
- Simulate 24 hours.

Example: Vagons V1.0 (answer)

```
        GENERATE    ,,,5
CICLE   ASSIGN      CARB,50
        QUEUE       INI
        SEIZE       CARG
MAS                 ADVANCE     6,1
        LOOP        CARB,MAS
        RELEASE     CARG
        DEPART      INI
        ADVANCE     360,120
        ASSIGN      CARB,50
        QUEUE       FIN
        SEIZE       DESC
MEN                 ADVANCE     13,3
        LOOP        CARB,MEN
        RELEASE     DESC
        DEPART      FIN
        ADVANCE     300,120
        TRANSFER    ,CICLE

        GENERATE    86400
        TERMINATE   1

        START       1
        END
```

Funavail

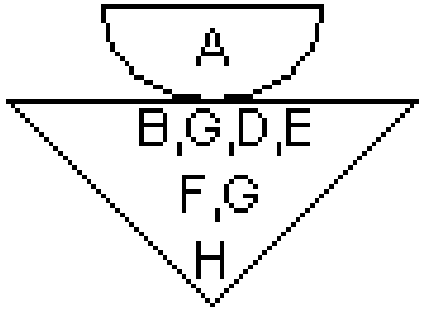
- Allows that an installation be available.
- A: name of the instalation.
- B: Modality. (op)
  - RE: remover.
  - CO: Continuar.
  - Nul.
- C: name of the bloc for the transaction that owns the instalation. (op)

Funavail

- D: number of the parameter that receives the residual time if the transaction is expulsed from the installation. (op)
- E: Modality de RE o Co. (op)
- F: name of the bloc for the PREEMP transactions of the installation. (op)

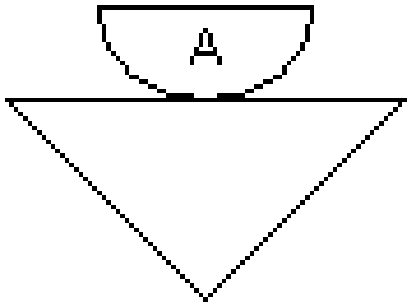
Funavail

- G: Modality RE o CO for the delayed transactions. (op)
- H: name of the new bloc for the pending transactions of the installation. (op)



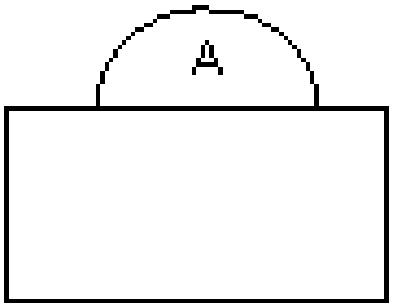
Favail

- Assures that an installation must be available.
- A: name of the installation



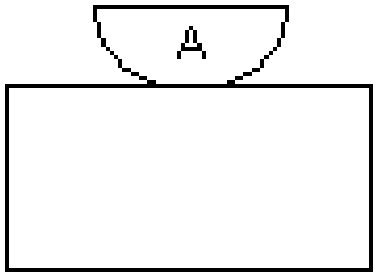
Sunavail

- Assure that the STORAGE is not available.
- A: STORAGE name.



Savail

- To guarantee that the STORAGE is available.
- A: STORAGE name.



Maquinat example

- New work every 10±4 minutes.
- Working time 15±5 minutes (2 identical machines).
- Simulate 1000 pieces.

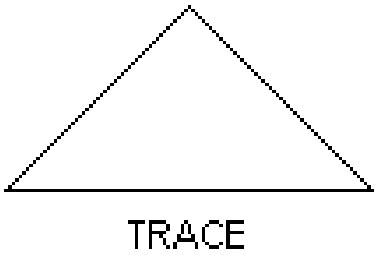
- Also…
- Every 90 minutes both 2 machines stops during 15 ±3 minutes.

Maquinat answer

```
STORAGE     S(MAQ),2

SIMULATE

GENERATE    10,4
QUEUE       INV
ENTER       MAQ
DEPART      INV
ADVANCE     15,5
LEAVE       MAQ
TERMINATE   1

GENERATE    90
SUNAVAIL    MAQ
ADVANCE     15,3
SAVAIL      MAQ
TERMINATE

START       3000
END
```

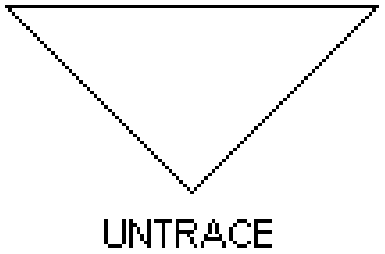
Trace

- Starts the trace of the transactions properties.
- The information that are printed are:
  - Number of the transaction.
  - Current block.
  - Destination block.
  - Clock value.



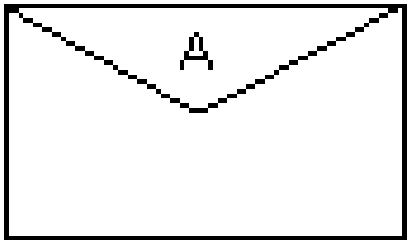
Untrace

- Stops the trace of the transaction properties.



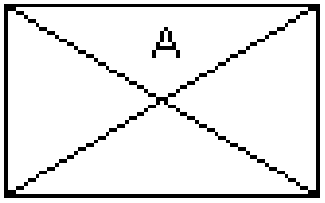
Assemble

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



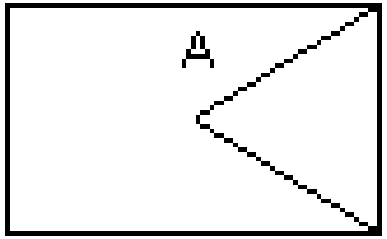
Gather

- To synchronize transactions.
- A: number of transactions we are waiting for.



Match

- To synchronize transactions.
- A: The other block MATCH.



Lathe example

- Pieces every 6 minutes.
- One worker, 3 phases of work.
  1. Lathe 3 minutes for piece.
  2. Take piece new dimensions (no time.
  3. Rectification 2 minutes piece.
- Recalibration of the machine that takes the dimensions for each piece 5 ± 3 minutes done by other worker. On the rectification the machine must be recalibrated.
- Simulate 200 pieces.

Tornejat answer

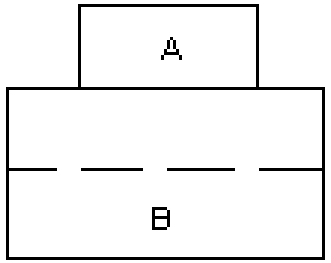
```
        SIMULATE

                GENERATE    6
                QUEUE               ALM
                SEIZE               OPER
                DEPART              ALM
                SPLIT               1,MED       Starts the calibration of the piece
                ADVANCE             3           Working
MED1            MATCH               MED2        Wait for the calibration of the machine
                ADVANCE             2
                RELEASE             OPER
                TERMINATE   1

MED             ADVANCE             5,3
MED2            MATCH               MED1
                TERMINATE

        START   200
        END
```

Modify the position of the transaction on FEC and CEC

- Modify the priority.
- Suspend the active transaction.
- Catch a machine, moving the transaction that owns it.

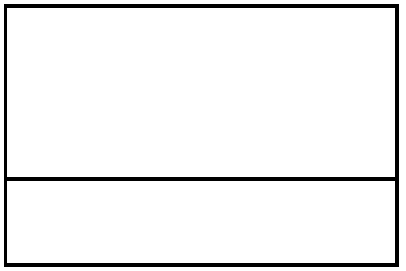
Priority

- Defines the priority over the active transaction.
- A: new priority value..
- B: Buffer option. (op). (see BUFFER).



Buffer

- Allows to reanalyze the CEC.

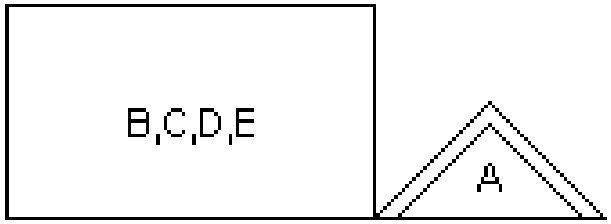


Preempt

- Displaces the transaction that owns the installation allowing that the new transaction takes it.
- A: installation id.
- B: priority mode. (op).
- C: identifier of the bloc for the moved transaction. (op).
- D: number of the parameter that receives the residual time. (op).

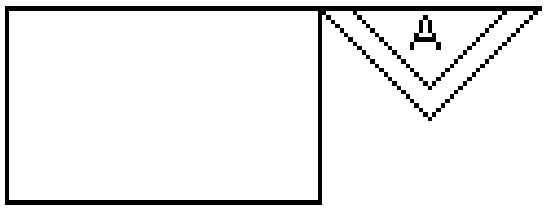
Preempt

- E: Remover modality. (op)



Return

- Free an installation that was been captured by a transaction.
- A: Installation name.

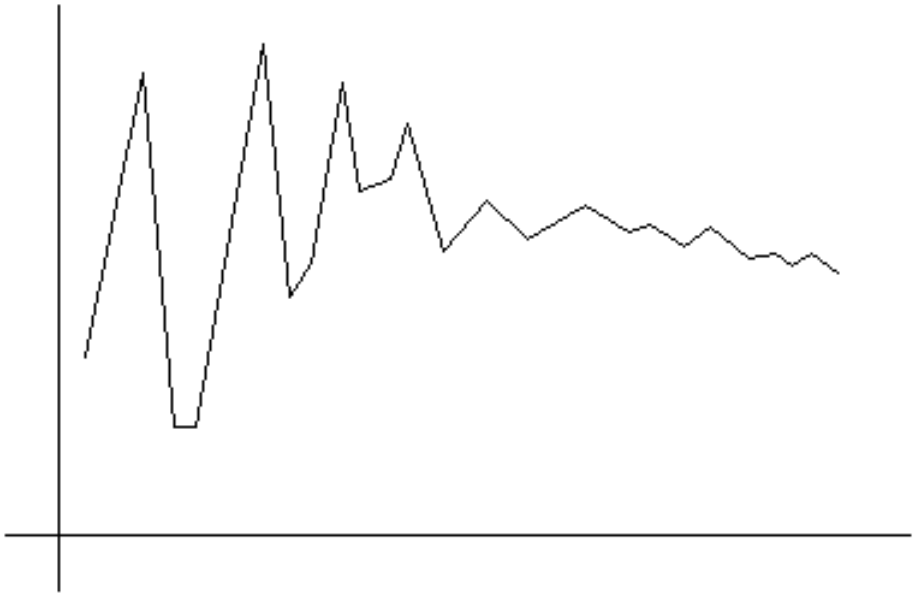


Replications (repetitions)

- To analyze the model n DIFERENT experiments are required.➔ Different behavior based on different RNG.
- To execute N different experiments.
- Between 3 an 10 replications. Calculus of the mean and the variance from these replications.

Transitory period

- Loading period of the model.



Clocks GPSS/H

- Relative clock: Time from the beginning of the simulation until the execution of RESET or CLEAR.
- Absolute Clock: Time from the beginning of simulation until the execution of CLEAR.

Reset

- RESET doe not delete the transactions of the current model.
- The blocs contents are not modified.
  - The statistic value total counts is set to current counts. (CLEAR put this to 0).
- Relative clock to 0.
- RNG are not initialized.
- Usually to define the loading period.

Clear

- Deletes all the transactions.
- Current counts and total counts set to 0.
- Clocks to 0.
- All the servers are set to free.
- RNG are initialized.

Clear

- Blocs contents set to 0.
- LOGICS set to 0.
- MATRIX elements set to 0.
- SAVEVALUES set to 0.

Transitory Time

- Time discarded from the statistics acquisition.

In/Out on GPSS/H

- File definition
- FILEDEF (control instruction)
  - OUT1 FILEDEF 'Sortida.txt'
- Inside the model the reference to the file is based in OUT1.

Getlist

- GetList FILE=nom,END=A,ERR=B,(&var1,&var2,..).
- Nom: logic name for the file to read the info.
- A: label of the control instruction if EOF is found.
- B: label of the instruction when read error is found.
- &var1: variable list to be read.[req].

Bputpic

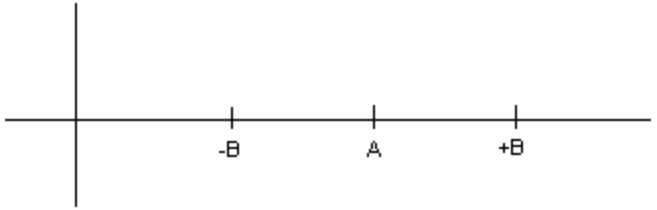
- To print the results.
- Name: name of the file.
- K: Number of text lines below the PUTPIC or BPUTPIC for the impressions of the results.
- A,B,C,..: Variable list, SNA, Savevalues, ampervariables to be printed.
  - PUTPIC FILE=SAL,LINES=2,(FC1,FC2).

Num. pieces machine 1: ******.

Num. pieces machine 2: ******.

Uniform distribution

- A,B ➔ A±B
- Inter Arrival Time = (A-B) + RN1*(2*B).
- RN1 on GPSSH is transparent for the user.



Distribució exponencial

- RVEXPO(j,IAT_{ave})
- IAT_{ave}= Temps mig entre arribades.

Poisson and Exponential

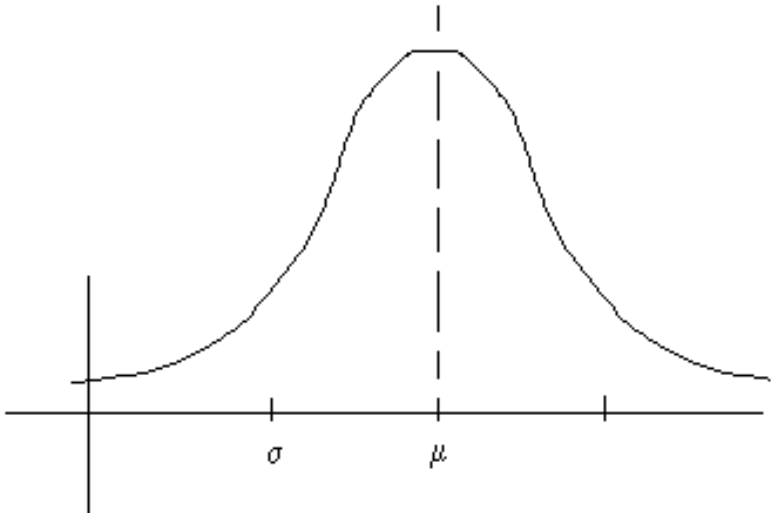
- Poisson→ n° of occurrences by interval. $\lambda=30$ arrivals/hour.
- Exponential→ Time between arrivals $\mu=1/30$ hours=2 minutes.

Erlang distribution

- (μ,K)=k exponencials de mitja μ/K
- ADVANCE RVEXPO(3,0.45)
- ADVANCE RVEXPO(3,0.45)
- ERLANG(0.9,2)
- ADVANCE RVEXPO(3,0.45)+ RVEXPO(3,0.45)

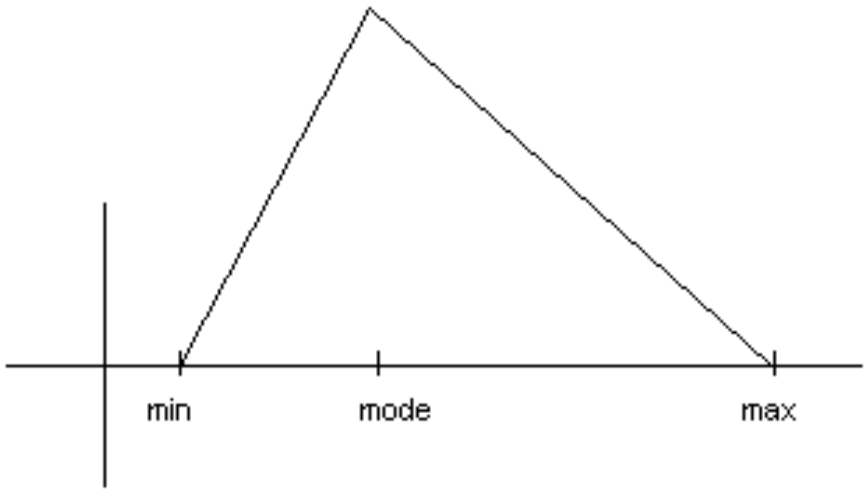
Normal distribution

- RVNORM(j, μ, σ)
- Be careful with the negative numbers ➔ ABS(RVNORM(j, μ, σ))



Triangular distribution

- RVTRI(j,min,mode,max)



Example of some distributions

```
    SIMULATE
*
*   ONE-LINE, SINGLE-SERVER QUEUEING MODEL
*
    GENERATE      1
    ADVANCE               5,2
    ADVANCE               RVEXPO(3,0.45)
    ADVANCE               RVEXPO(3,0.45)+RVEXPO(3,0.45)
    ADVANCE               ABS(RVNORM(1,5,3))
    ADVANCE               RVTRI(1,2,3,5)
    TERMINATE     1
*
```

Internal vision of the transactions 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 esdeveniment chain→always1
2. Future esdeveniments chain→ always1 1
3. Users chain→0 or more
4. Interrupt chain→0 or more
5. MAQ chains→0 or more

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

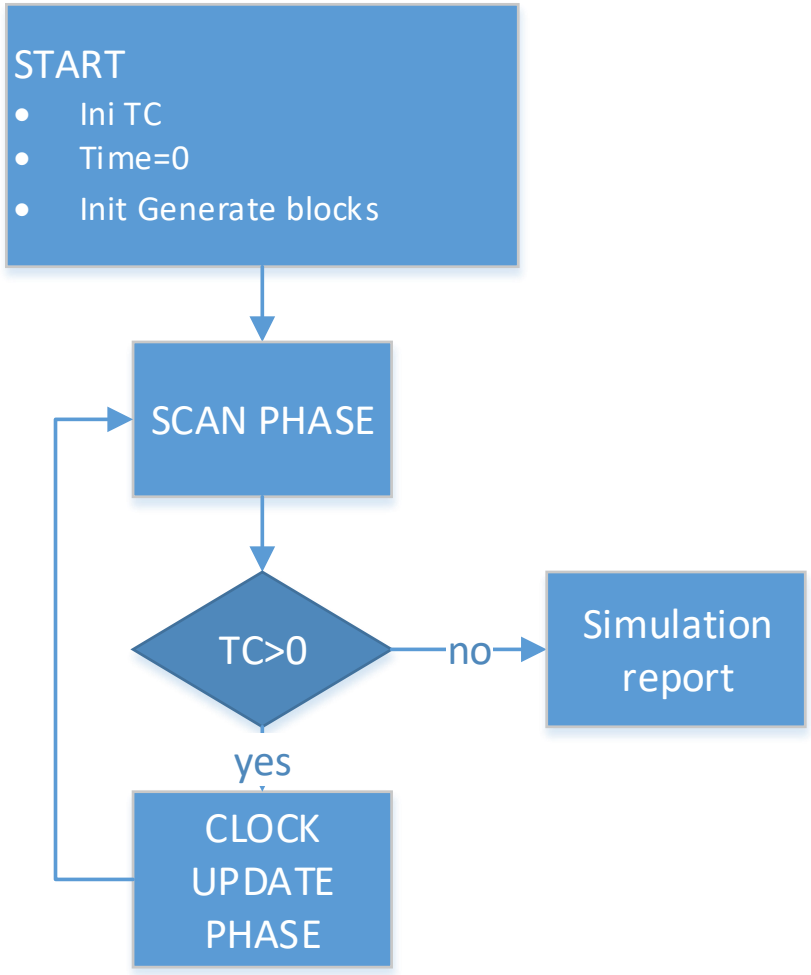
CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
3	0	480.000	1	4	5		

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

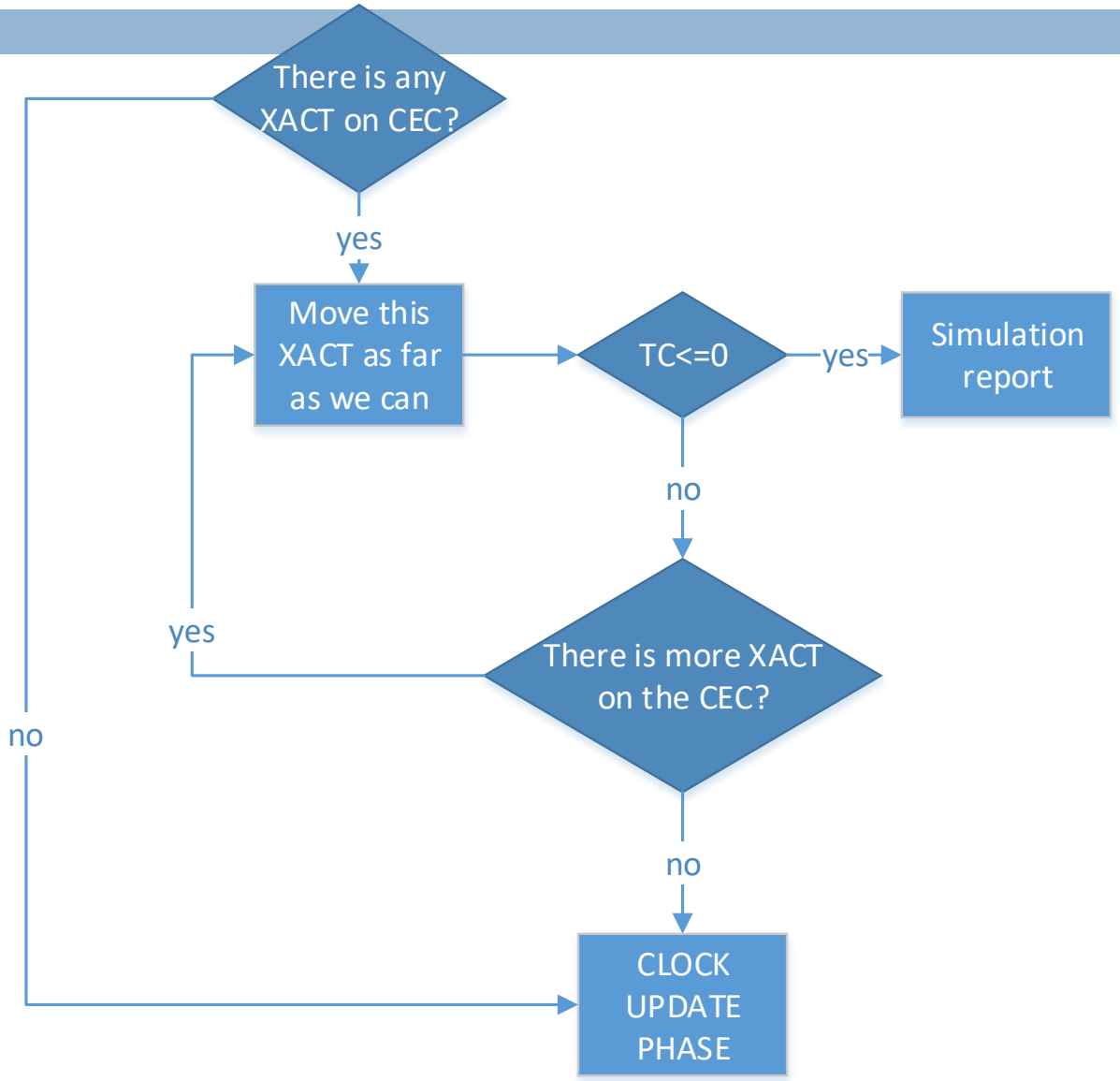
GENERATE blocs initialization

- 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

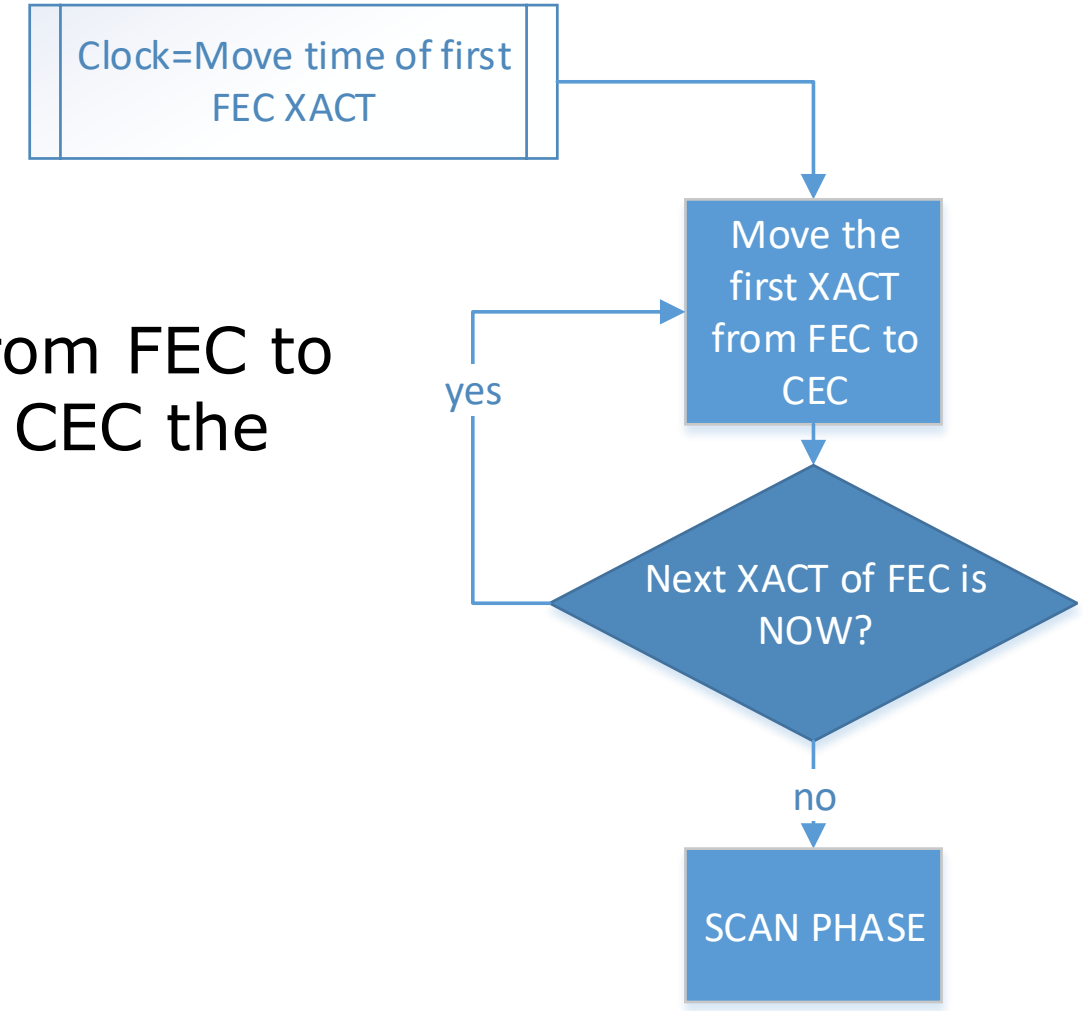


SCAN PHASE


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.



```
//SCAN PHASE.
        while (CEC.size() > 0 && TC > 0) {
            xact = (Xact) CEC.remove(0);
            Bloc aux = xact.getBloc();
            while (aux != null) {
                aux = aux.execute(xact);
            }
        }
```

```
//TODO 4: CLOCK UPDATE PHASE
        if (FEC.size() > 0) {
            xact = (Xact) FEC.get(0);
            relativeClock = xact.getMoveTime();

            while ((FEC.size() > 0)) {
                if (((Xact) FEC.get(0)).getMoveTime() != relativeClock) {
                    break;
                }
                CEC.add(FEC.remove(0));
            }
        }
```

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.