**GRUPO 1**

Xabier Jiménez

Josune Sorbet

Rubén Chérif

**PERSONAL 2 (INDIVIDUAL) 12/09/2019-18/09/2019**

Preguntas de la parte número 1:

-Se pueden generar entidades dada una función probabilística y se puede decidir a qué conector (o conectores) es desviada si existe más de uno ¿Pero se puede añadir una probabilidad a la salida que puede tomar una entidad?

Preguntas de la parte número 2:

-Diferencia entre batch y split? Es que en el split se tienen que volver a unir en el join sí o sí?

Preguntas de la parte número 3:

*“The Gate Activity accumulates Entity instances until some number of Entity instances have been received or until a signal is received to release Entity instances.”*

*Menciona que acumula entidades hasta que le llegue una señal; como es posible definir dicha señal y donde habría que definirla? Mediante programación dentro de la actividad?*

Mediante MediThreshold Release se puede establecer un número y cuando el número de instancias de entidades acumuladas sea igual o mayor, se liberan una cantidad de entidades. No encuentro donde es posible determinar el número de entidades que serán liberadas en ese caso. Supongo que mediante programación se puede hacer, pero quisiera saber si existe otra alternativa.

*Si se definen n pads y por tanto, tenemos n colas se le puede especificar a cada un método de ranking distinto? O todas las colas del mismo objeto deben seguir el mismo? Por ejemplo la cola 1 donde llegan entidades tipo uno podrian tener FIFO y la cola 2 donde llegan entidades tipo 2 tener First In Last Out?*

**SESIÓN PRÁCTICA 19/09/2019**

**EJEMPLOS DE LA PRIMERA PARTE:**

<https://drive.google.com/open?id=1luJ5l3N-vxYcPAWkDHixbp31FpQvIIS3>

**Generate:** Crea una o varias entidades dada una distribución estadística o un valor constante.

**Delay:** Las entidades que entran esperan una cantidad de tiempo fija o una definida por una distribución estadística.

**Dispose:** Elimina entidades. Útil para recoger valores estadísticos. Se puede definir un número máximo de unidades que no puede exceder.

**Assemble:** Combina entidades para crear otras nuevas. Tienes 4 conectores: uno por el que entran las entidades (component), otro por el que entran los triggers (triggers), otro por el que salen las entidades creadas (out) y el último, por el que salen las entidades con las que el assemble no puede hacer nada (no match). Los triggers son funcionalmente iguales a las entidades combinables y se pueden activar/desactivar marcando “hold for trigger”. Se puede definir un delay al proceso.

**Branch:** Redirige las entidades que le llegan por un único conector a varios distintos.

**Merge:**  Lo contrario a un branch. Junta varios conectores en unos solo. No hace distinciones entre entidades.

**EJEMPLOS DE LA SEGUNDA PARTE:**

**Batch**: Agrupar por cantidad, no se puede agrupar siempre un azul con un negro o así. En una empresa creas varias piezas por separado y en un momento dado las tienes que empaquetar todas para el transporte, a partir de ese momento todas las piezas son tratadas como un único elemento.

**Unbatch**: Al llegarte la caja llena de piezas las desempaquetas y las vuelves a tratar como “cosas” distintas (complementario del batch)

Retain Batched Entity:Para que salga la caja

Unbatch Nested Batches: Si es una caja de cajas que las cajas de dentro también se abran

Ejemplo batch umbatch:

<https://drive.google.com/open?id=1UExPbRTbP0jc5yC5W5tJuhmPkIqEJoSO>

**Assign**: Sirve para asignar valores a los atributos y para cambiar la prioridad de una entidad es decir: si estás siguiendo el curso de los lapiceros de tu empresa que reparte a un departamento de arte y a otro de ingenieros, la prioridad para los lápices en arte es mucho mayor que en ingeniería por lo que habría que asignarle un valor diferente para cada departamento

**Transform**: transforma el tipo de la entidad en otro tipo en varias entidades de este nuevo tipo. Tienes productos que vendes, al venderlos ya no te interesa tener la entidad producto, si no que te interesa tener la entidad dinero, así que lo transformas en esa nueva entidad.

Ejemplo assign transform: <https://drive.google.com/open?id=1liH6dKLCvkEhvOfc415iI9ZYD0iNLpvS>

**Split**:Si tienes un conjunto de empresas realmente cada empresa está divida en diferentes actividades como: contabilidad, rrhh,... por lo que interesa tener todas estas actividades por separado pero que al final se trate como una única empresa

**Join**: eliges cuales quieres que se unan de nuevo, es decir igual te interesa saber cuál es la empresa pero también quiéres saber el dinero recaudado: te quedarías con el padre(la empresa) y con uno de los hijos y “matarías” a los demás. (Es el complementario del split)

Ejemplo split join:

<https://drive.google.com/open?id=1pTS3mQwNe-vqfXSw_cPke_fjhTZH7Oy1>

**TERCERA PARTE:**

Gate: Nos permite acumular un determinado número de instancias de entidades. El gate dispone de dos pads(entradas),:

* Hold Pad: es una cola para las entidades que llegan. En cuanto una entidad llega, se encola al final de la cola.
* Trigger Pad: cuando un trigger llega envía una señal para que las entidades se lanzan. Es similar a que cuando llegue un trigger la puerta se abra y las entidades puedan salir del gate.

Para que las entidades sean lanzadas es necesario que se envíe una señal mediante un trigger, dichas señal será lanzada cuando se cumpla una de las siguientes políticas:

* Threshold Release: nos permite establecer cuantas entidades serán lanzadas una vez alcanzado un número N de entidades acumuladas en el gate. Por tanto, cuando en el gate haya N entidades acumuladas se lanzará una señal (trigger) y un número de entidades serán lanzadas.
* Trigger Release: Nos permite

Synchronize Activity: permite coordinar el lanzamiento de las diferentes entidades que llegan. Para ello, se le podrá indicar el número de entradas/salidas del que dispondrá. Las diferentes entidades que llegan son encoladas en cola correspondiente, dependiendo de por donde hayan llegado.

Propiedad Duration: si el valor se establece como “none”, una vez llegue una entidad de cada camino, dichas entidades serán lanzadas con el mismo orden con el que llegaron, pero nunca podrán ser lanzadas antes de que llegue al menos una de cada tipo.

Si el valor se establece como X>0, desde que llegue la primera entidad de cualquier tipo hasta que se llegue al tiempo establecido X no se lanzará ninguna entidad y mientras tanto las entidades que vayan llegando se irán acumulando cada una es su cola correspondiente. Una vez cumplido el tiempo todas las entidades acumuladas serán lanzadas en el mismo orden de llegada.

Esta actividad nos permite establecer la política de colas, pero todas las colas deben de seguir la mismo política: FIFO, First In Last Out, Earliest Created First, Latest Created First, Highest Priority Value First, Lowest Priority Value First, Highest Attribute Value First. Lowest Attribute Value First.

Get Resource: cuando una entidad pase por esta actividad se le dará X cantidad del recurso asignado. Si dicho

Free Resource: permite liberar los recursos asignados por un Get Resource.

Release all Allocated Resources: libera todos los recursos asignado a la entidad que venga.

List of actions: nos permite indicar qué recursos asociados a diferente Get Resources serán liberados. Por ejemplo si una entidad ha pasado por 3 diferentes Get Resource y cada uno le ha asignado un recurso diferente se le podrá indicar que solo libere el recurso 1 y 3.

Si un recurso es consumible dicho recurso no podrá ser liberado.

Replenish Resource: se activa cuando una entidad llegue y permite reponer un recurso.

List of replenish actions: Se le puede indicar qué recursos y la cantidad a reponer.

Replenish to capacity: repone la cantidad de un recurso al inicio. Si una recurso al inicio ha sido definido con 20 unidades disponibles, se repondrá 20 unidades

Ejemplo:

<https://drive.google.com/drive/folders/14HtjneuOUj8SSvS_GBhpxNNtoL3rNBHp>

**REPARTO 4:**

**Distribución uniforme:**

Uni(minimum, maximum, stream) o UniACM(minimum, maximum, stream)

Misma posibilidad de que “caiga” cualquier número del intervalo entre minimum y maximum

**Binomial:**

Bin(trials, probability, stream) or BinACM(trials, probability, stream)

El experimento se repite tri

als veces y los resultados son independientes entre sí.

Los éxitos ocurren con una probabilidad p y los fracasos con una probabilidad de 1-p

**Exponencial**

Exp(mean, stream) or ExpACM(mean, stream)

La media tiene que ser mayor que 0

**Beta**

Bet(shape1, shape2, minimum, maximum, stream) or BetACM(shape1, shape2, minimum, maximum, stream)

minimum >= 0.0, maximum >= 0.0, and maximum > minimum

shape1 >= 0.0 and shape2 > 0.0

Está torcida hacia la derecha o hacia la izquierda en función de los valores que se le de a shape1 y a shape2

**Background text:**

S**tatic**: sale hasta que suceda la condición para que me muestre el dinamic

**Dinamic**: pones una condición y te va mostrando por pantalla lo que le pidas.

Para que se muestre por pantalla el dinamic hay que darle al siguiente botón:



**Resources en el delay:**

El delay le asigna y le retira el resource. Se puede poner consumible y demás características de los delays.

Ejemplo:

<https://drive.google.com/drive/folders/1UHvctP0lvzYA-LrIP0wymxrXcoxfVvRP>

**EXPLICACIÓN REPARTO 5:**

**INTERVALOS DEL GENERATE**

Se puede definir más de un intervalo de tiempo en la actividad generate. Los intervalos a su vez pueden ser de distinto tipo dependiendo del espacio de tiempo y las características de su distribución : Periodic, cyclical, yearly, monthly… De esta forma podemos, por ejemplo, definir dos intervalos “daily” uno para turnos de mañana y otro para turnos de noche, que se vayan intercalando según especifiquemos, y que tengan dos intervalos distintos.

**RECURSOS**

Como parte de la definición de una actividad, se le puede indicar si necesita un tipo de recurso para ejecutarse. Si es accesible inmediatamente, lo reservará, haciéndolo inaccesible para otras actividades y los liberará ( o destruirá, dependiendo de la especificación de los recursos) tras haberse ejecutado. Si no son accesibles, esperará hasta que pueda reservarlos.

La reserva de recursos sigue una cola FIFO con varios niveles de prioridad (la prioridad se especifica en las actividades). Es decir, actividades de mayor prioridad accederán antes a los recurso que otras declaradas como menos importantes.

A la hora de definir recursos, se pueden especificar:

-Nombre: único en cada recurso, no se pueden usar caracteres como / ó \*.

-Unidades: se puede definir el número de unidades iniciales.

-Uso fraccionado: si se pueden reservar unidades no enteras de un recurso.

-Consumible: y si desaparecen o no tras su uso .

-Coste: que va asociado al uso del recurso.

-Expresiones, atributos, documentación, metadatos, comentarios y ayuda…

Se pueden definir el comportamiento de las actividades con respecto a su reserva de los recursos:

-Any one member: coge un/unos recurso cuales quiera de los habilitados

-All members: reserva todo los recursos de golpe cuando están libres.

-Reserve as available: va reservando los recurso conforme se van liberando **PUEDE PROVOCAR DEADLOCK**

**-**Number of members: se necesita un número especificado de recursos

Se puede definir un tiempo máximo de espera, tras el cual, si la actividad no se ha ejecutado, muere (el atributo endWait se declara como TRUE). Útil para reunir datos estadísticos.

También se puede liberar recursos durante la ejecución o interrupción de una actividad, sin necesidad de que esta haya terminado.

**REPARTO 6:**

Atributos: existen dos tipos de atributos:

* Definidos por el sistema
* Definidos por el usuario

A veces los atributos del sistema no aportan toda la flexibilidad necesaria a la hora de modelar por tanto el propio usuario puede definir nuevos atributos con el fin de poder modelar situaciones complejas:

Ejemplo: en un batch se pueden acumular entidades y la condiciones que se debe cumplir es que cuando haya X entidades acumludas, entonces se liberen. Sin embargo, si desearamos un modelo en el que las entidades fueran lanzadas dependiendo del peso no podríamos hacer nada. Por ellos, la solucion seria crear un nuevo atributo que contuviera el peso de las entidades y de esa forma poder determinar cuándo serán lanzadas.

Otra de las situaciones en las cuales los atributos serían necesarios seria a la hora de recoger estadísticas sobre un aspecto en concreto, por ejemplo, calcular cuántas unidades de peso > X han sido procesadas.

El valor de los atributos podría ser variable en el caso de que por ejemplo el valor dicho atributo dependiera de factores dinámicos.

Funciones principales atributos:

* Alterar comportamiento de los procesos mediante cambios en el valor de los atributos
* Información de comunicación entre procesos
* Almacenar información durante la simulación

Dentro de los atributos del sistemas existen dos tipos:

* Get-Set: su valor puede ser modificado (atributos de control)
* Get-Only: su valor no puede ser modificado (informacion estado simulación)

Atributos Globales: entidad, tipo entidad, recurso, actividad, modelo

Los atributos definidos a las entidades son definidos por cada instancia de las entidades, eso quiere decir que que cada instancia tendrá un atributo independiente del las otras instacias de las entidades.

Sin embargo, los atributos definidos para los tipos de entidades, existe un atributo para todas las instancias de ese tipo de entidad.

Expresiones:

En las siguientes paginas se muestra cosas sobre el lenguaje:

* Tipos de datos: BOOLEAN, REAL, INTEGER…
* Operadores:
  + Asigancion: ‘:=’,
  + Logicos: ‘=’, ‘<>’, ‘<’, ‘>’, ‘<=’, ‘>=’
  + Aritmeticos: ‘+’, ‘-’, ‘\*’, ‘/’, ‘MOD’

- Literales: str = “myString”;, muBool = TRUE;...

- Sentecias:

IF condition

{ ELSIF condition}

{ ELSE }

END IF;

SWITCH value

{ CASE value}

{ END CASE; }

{ OTHERWISE }

END SWITCH;

WHILE condition

END WHILE;

FOR expression TO | DOWN TO expression [ BY expression]

END FOR;

RETURN (possible return value)

EXIT (with WHILE or FOR loops)

**Atributos en Expresiones:**

* Referirnos a atributos de una entidad: *Entity.DefaultAttribute*
* Atributo Modelo: *Model.DefaultAttribute*
* Atributo de un recurso: *Resource("Resource1").UnitsBusy*

Expression Activation Events:

* SIMPROCESS tiene un conjunto de activacion de eventos en donde se encarga de comprobar la existencia de las expresiones y posteriormente ejecutarlas.
* Esto se lleva a cabo varias veces durante la simulación. Los eventos Start Trial y End Trial ocurren una vez por cada replicación. Con los eventos StartSimulation y End Simulation pasa exactamente lo mismo siempre que en los ajustes de ejecución se selecciona la opción Reset Settings, si no, dichos eventos sólo serán ejecutados una vez durante una simulación con varias replicaciones
* Start Run and End Run son expresiones del modelo y siempre son ejecutadas una sola vez durante la simulación.
* Las expresiones como ‘Accept Entity’ que se encuentra dentro de una delay por ejemplo, pueden ser ejecutadas múltiples veces (tantas como entidades lleguen).
* El valor del los atributos de una entidad solo puede ser definido una vez sa cree una instancia(Initialize Entity) o Release Entity. Sin embargo, no se puede definir al inicio de la simulación, debido a que todavía no se han generado instancias de las entidades.

Funciones (dev):

Devuelve el valor de la expresión cada vez que se llama a la funcion. Nos permite llamar a ala función desde diferentes sitios durante la simulacion. Por tanto la ultima sentencia de la expresion debe ser ‘return X’.

Permite el paso por argumentos, para ello, los nombres deben ser Ar g1, Ar g2,..., Ar gN y deben ser definidas en orden con su respectivo tipo (INTEGER, REAL, …).

Métodos (no dev):

Es lo mismo que una función pero no devuelve nada. Desde la pagina 55-90 se muestras todos los métodos definidos por el sistema, muchos de ellos son demostrados con modelos el directorio ‘ExpressionDemos’.

Ejemplo: [Ejemplo Reparto 6](https://drive.google.com/drive/folders/1wauUqKVl2ZodNM_V99j00AV_2goumSnW)

**Versiones Proyecto**

* **Lectiva 8.2 - 24/10/2019**
  + [Versión 1](https://drive.google.com/drive/folders/1rMn83z_p4R2S7cKDOvnOinmDnm19DSpe)
  + [**Versión 2**](https://drive.google.com/drive/folders/1BcP74UnmejohvXEKoRuflM5iHTQhrgID)
* **Lectiva 9.2 - 30/10/2019**
  + [Versión 1](https://drive.google.com/drive/folders/1ANHo7VgeWxr92i3TdEatwV47P7zRoGDN)
  + [Versión 2](https://drive.google.com/drive/folders/1xLy9zGV3Vlq2BkyBc9jvU3w3to96Kw42)
  + [**Codigo Version 2**](https://drive.google.com/drive/folders/1SJkGPU2yZZcc4VqOUtXzmDB9m0_12Yfb)
* **Lectiva - 6/11/2019**
  + [**Versión1**](https://drive.google.com/drive/folders/1d_xZxdlDs2Tf4SgMIsKQGGjULB2xwGKz)
* **Lectiva- 7/11/2019:**
  + Contador de nodos libres, y cambiado el Acept (solo llama al trigger si la cola está vacía) y Release entity del Gate. Creado un Delay que libera nodos y canales una vez salen los procesos terminados (el delay espera un tiempo fijo, no el tiempo de ejecución). Diseñados algoritmos de captación de nodos (a falta de implementarlos).
  + [Version1](https://drive.google.com/drive/folders/1J0smJhRz6P-WbWhycnF2e_TjF5roV3xs)
  + [Version2](https://drive.google.com/drive/folders/1Pn8BaOYB-oE5AIP8Lp6Qh8UBvw5BqGNt)
  + [**Versión3**](https://drive.google.com/drive/folders/17A6VsOxmZVCCYEYMy3XagNlF3BqvntBo)
* **Lectiva 13/11/2019:**
  + Implementación del algoritmo asignar nodos y las funciones a las que llama.
  + **Faltan por asignar los canales!!!!**
  + [**Versión1**](https://drive.google.com/drive/folders/1Nq2qcfxIujSE-CSl4O5hjGcef2PQNwGo)
* **14/11/2019**
  + [VERSION1](https://drive.google.com/drive/folders/1gxzvap_67ip43CuEhvaZb4PJD8k1t7L2): Al llegar un programa al disposers, éste comprueba si el siguiente programa en la cola se puede ejecutar y manda crear un triggers si es posible.
  + [Modelo 2](https://drive.google.com/drive/folders/1SZrNRVCyfsVYGlNKVTX7m7pGQ2H4uxcb): Creada la primera versión del segundo modelo del sistema con dos colas y acceso restringido a la estructura de procesadores. El modelo cuenta ahora con dos puertas, dos delays, dos generadores de triggers y dos disposers distintos. La estructura del código sigue siendo la misma, aunque se han sustituido referencias de atributos viejos por otros actualizados.
  + [Version2](https://drive.google.com/drive/folders/13AGBLBjO8JxBUaMRQ4XTbOUHAENlbDp8): Corregidos aproximadamente 149.293.028 errores del código de asignación de nodos. Cambiado el sistema de asignación de nodos, ahora se van reservando conforme se encuentran, a diferencia de antes, que se reservaban una vez se había encontrado una secuencia conexa.

A CORREGIR: Hay que definir nDerecha y nIzquierda como variables globales para que no se reinicien con el comienzo de cada nueva iteración.