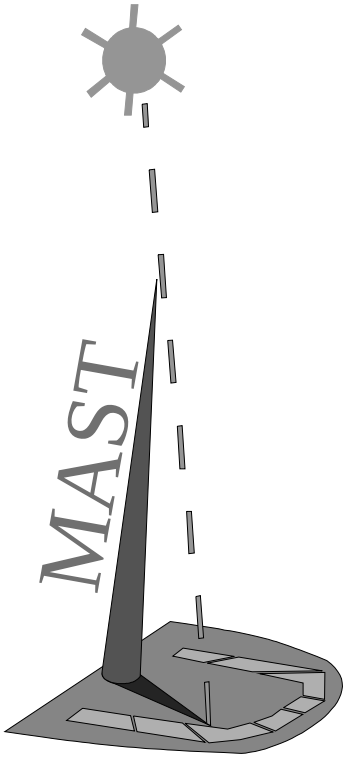
GRUPO DE COMPUTADORES Y TIEMPO REAL

UNIVERSIDAD DE CANTABRIA

UML_Mast_Metamodel

(Versión 1.0)



José M. Drake
Julio Medina
Santander, febrero, 2001

INDICE

LISTA DE DIAGRAMAS

UML MAST METAMODEL: INTRODUCCIÓN.

UML Mast es una metodología y un conjunto de herramientas gráficas desarrolladas para modelar y analizar sistemas de tiempo real que están siendo desarrollados utilizando métodos orientados a objetos sobre herramientas CASE basadas en la notación UML. Los componentes de modelado y las herramientas de análisis proceden del entorno MAST (Modeling and Analysis Suit for Real Time Aplications).

El modelo de tiempo real del sistema que se desarrolla se formula mediante la "Mast RT View". Esta es una vista complementaria de la descripción UML del sistema que modela la capacidad de la plataforma que se utiliza, las características de temporización de los componentes lógicos de su software y las transacciones que pueden ocurrir con los requerimientos temporales que en ellas se establecen. Con la Mast RT View el diseñador puede construir gradualmente el modelo de tiempo real del sistema que diseña, de forma paralela al desarrollo de su modelo lógico

El metamodelo constituye la descripción básica y formal de la Mast RT View. En este documento se describe con notación UML este metamodelo y con él se definen:

1. Se define la estructura general del modelo, estableciendo sus secciones, los aspectos del sistema que describe y los componentes con que se construyen.

2. Se describen los componentes que se utilizan para construir el modelo de tiempo real, definiendo su naturaleza así como los atributos que describen su comportamiento cuantitativo.

3. Se describen las relaciones que se pueden establecer entre los componentes dentro del modelo.

4. En el modelo, a través de estereotipos se indica el tipo de componentes UML (paquete, clase, atributo, actividad, acción, evento, etc.) con el que se instancia cada componente del metamodelo en el modelo concreto de un sistema.

La descripción del metamodelo se realiza a través de diagramas de clases que describen gráficamente su estructura, y de párrafos de texto que describen conceptualmente cada componente así como sus atributos. A fin de minimizar dependencias cruzadas en la descripción de cada diagrama, se incluye junto a cada uno la mayoría de las descripciones de sus clases, aunque ello implique la repetición de clases a lo largo del documento.

Del metamodelo existe una descripción equivalente a la de este documento en formato ROSE "UMLMast_Metamodel.mdl"

UML Mast Metamodel

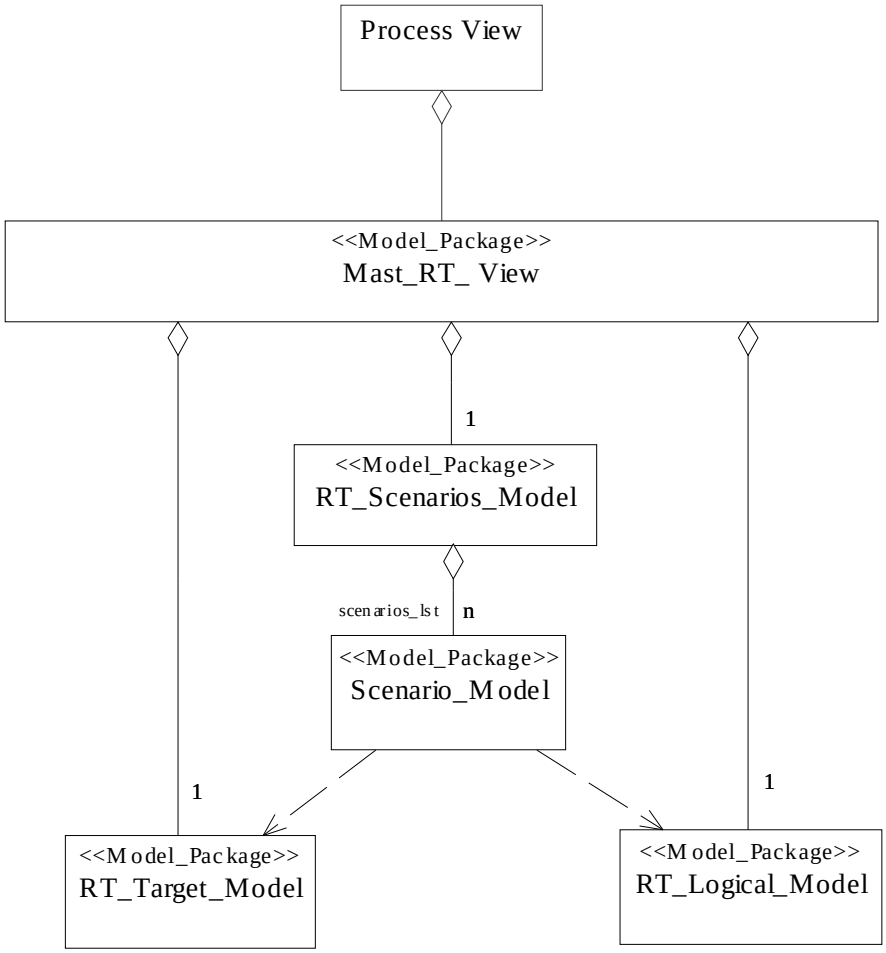


Diagrama 1.- Mast RT View.

Process View

Es una de las vista del "paradigma 4+1" que se usa para describir la performance, escalabilidad, y throughput de un sistema bajo desarrollo en una herramienta CASE UML.

La vista Process View no se utiliza en la herramienta ROSE'2000, por lo que sólo se define en el metamodelo de forma conceptual. En el modelo la Mast_RT_View se declara en el package Logical View ya que aunque conceptualmente no esta relacionada con él hace uso del mismo tipo de componentes UML.

Mast_RT_ View

Es una vista adicional que complementa la descripción estándar UML de un sistema durante su fase de desarrollo. Su finalidad es definir un modelo de tiempo real que describa la temporización y los recursos que se asignan a las actividades del sistema y sirve de base para analizar su planificabilidad mediante herramientas automáticas.

Se implementa como el paquete raíz dentro del que se definen todos los componentes del modelo de tiempo real. Este paquete debe estar definido directamente sobre la Logical View a efecto de que pueda ser localizada por la herramienta automática.

RT_Target_Model

Modela la capacidad de procesamiento y las restriciones operativas de los recursos de procesamiento hardware y software que constituyen la plataforma sobre la que se ejecuta el sistema.

RT_Logical_Model

Modela los requerimientos de procesado que requiere la ejecución de las operaciones funcionales definidas en los componentes lógicos que se utilizan en el diseño, tales como métodos, procedimientos y funciones definidos en las clases, primitivas de sincronización entre threads, sesiones de comunicación, etc.

Describe el comportamiento de tiempo real de los componentes funcionales (clases, métodos, procedimientos, operaciones, etc.) que están definidos en el sistema y cuyos tiempos de ejecución condicionan el cumplimiento de sus especificaciones de tiempo real.

El RT_Logical_Model modela dos aspectos de un componente:

- La complejidad de los algoritmos con que se implementa.
- Las posibilidades de bloqueo que pueden retrasar su ejecución y que son consecuencia de tener que hacer uso exclusivo de recursos compartidos (Shared_Resource).

RT_Scenarios_Model

Conjunto de configuraciones o de modos de operación del sistema en los que hay definidas especificaciones de tiempo real.

Se implementa como un paquete que debe estar definido directamente dentro del paquete Mast_RT_View.

Scenario_Model

Modela cada configuración o modo de operación que puede ser alcanzado por el sistema en el que estén establecidos requerimientos de tiempo real.

Representa una carga concreta de trabajo (workload) para la que deben satisfacerse un conjunto de requerimientos de tiempo real. Está constituido por el conjunto de los modelos de las transacciones que concurren en él.

En un modelo cada Scenario_Model se representa mediante un package con el nombre del escenario y se define en el package RT_Scenarios_Model.

Metamodel Generic Types

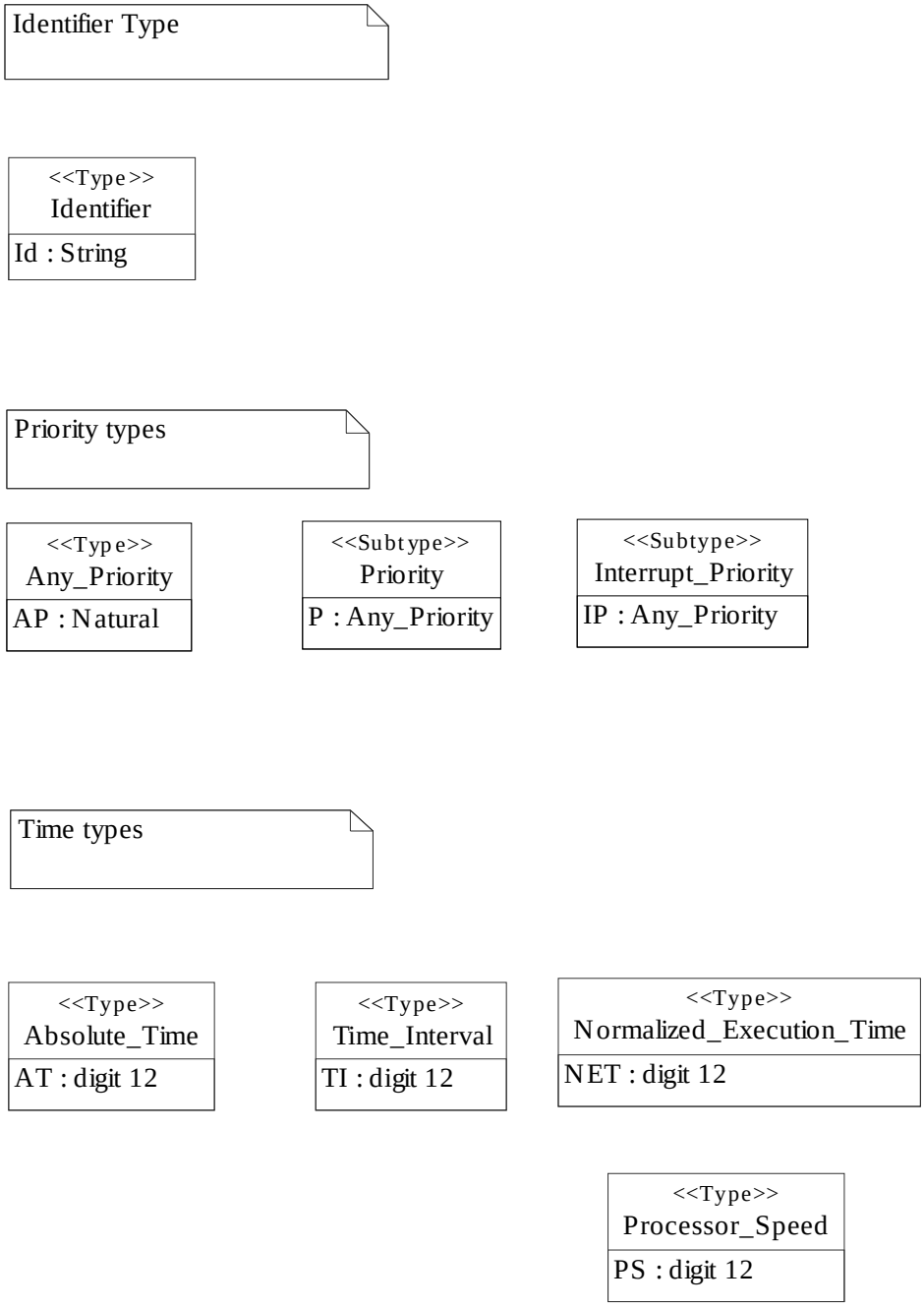


Diagrama 2: UML-Mast types.

Identifier

Identificador de un elemento del modelo Mast. Identifiers folow the Ada rules for composite identifiers: they begin with a letter ('A'..'Z'|'a'..'z') and they are followed by letters, digits ('0'..'9'), underscores('_') or periods('.').

Private Attributes:
Id : String

Any_Priority

Nivel de prioridad de un scheduling server o de una actividad. Proporciona unas escalas absolutas de prioridades válidas tanto para la planificación de las actividades dentro de un mismo recurso de procesamiento, como para la planificación del acceso a los recursos de un mismo procesador (recursos locales) o de diferentes procesadores (recursos globales).

Private Attributes:
AP : Natural

Priority

Niveles de prioridad que pueden ser asignadas a los scheduling server de nivel de aplicación. El rango de este subtipo es dependiente de cada procesador, y es definido en su declaración.

Private Attributes:
P : Any_Priority

Interrupt_Priority

Niveles de prioridad que pueden ser tomados por los scheduling server en que se ejecutan tareas de atención a las interrupciones hardware. El rango de este subtipo es dependiente de cada procesador, y está definido en su declaración .

Private Attributes:
IP : Any_Priority

Absolute_Time

Tiempo absoluto expresado en segundos. Un tiempo absoluto está expresado respecto de un origen de tiempo indefinido, pero que es común para todos los Absolute_Time definidos en el modelo.

Private Attributes:
AT : digit 12

Time_Interval

Intervalo temporal expresado en segundos.

Private Attributes:
TI : digit 12

Normalized_Execution_Time

Tiempo normalizado de ejecución de una operación.

Real execution time = Relative execution time / Speed_Factor

Private Attributes:
NET : digit 12

Processor_Speed

Tipo que expresa el Speed_Factor de un Processing_Resource. El tiempo de ejecución de una operación se obtiene dividiendo el tiempo normalizado por el factor de velocidad del procesador. Los tiempos de ejecución de las operaciones son expresados como los tiempos que tardaría su ejecución en un processing_resource con Speed_Factor=1.0.

Private Attributes:
PS : digit 12

Target Metamodel

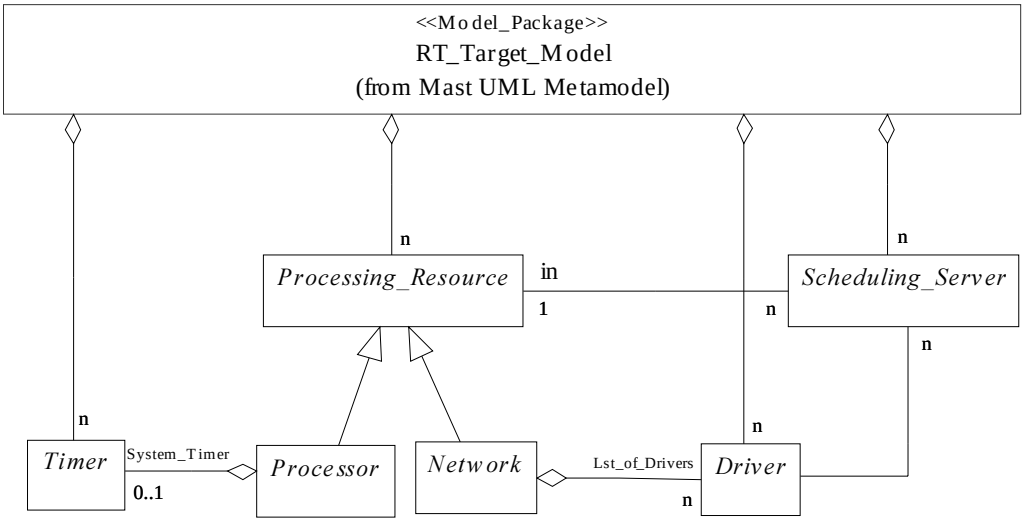


Diagrama 3: RT_Target Model.

RT_Target_Model

Modela la capacidad de procesamiento y las restricciones operativas de los recursos de procesamiento hardware y software que constituyen la plataforma sobre la que se ejecuta el sistema.

Processing_Resource

Modela un componente que ejecuta actividades que son parte del sistema que se modela.

Modela los componentes hardware y el software de background (sistema operativo, drivers, etc) en los que se ejecutan las actividades del sistema.

Private Attributes:

Speed_Factor : Processor_Speed

Factor de velocidad de procesamiento del recurso. El tiempo real de ejecución de cualquier actividad que se ejecute en el recurso se obtendrá dividiendo el tiempo de ejecución(Worst_Case_Execution_Time,Avg_Case_Execution_Time, Best_Case_Execution_Time, etc.) establecido en las operaciones que lleva a cabo, por el factor de velocidad.

Processor

Recurso de procesamiento que ejecuta código de la aplicación. Su capacidad de computo se distribuye entre la ejecución de código de las actividades del sistema que tiene asignadas y la ejecución de las tareas de gestión, monitorización y cambio de contexto entre actividades.

Derived from: Processing_Resource

Network

Recurso de procesamiento especializado que modela la ejecución de las operaciones de transferencia de mensajes entre procesadores a través de un mecanismo de comunicación existente en la plataforma. Modela el tiempo finito que requiere transferir un mensaje entre dos procesadores y la contención que produce en los procesos que realizan la transferencia el hecho de que deben realizarse uno a uno con exclusión mutua.

Derived from: Processing_Resource

Scheduling_Server

Modela un proceso activo simple (con un solo thread) dentro del que se ejecutan una o varias actividades secuencialmente y que es ejecutado por el recurso de procesamiento que tiene referenciado.

Así mismo, modela la política de planificación y la capacidad de procesamiento disponible para ejecutar las actividades que se le asignan.

Derived from: Job_Parameter_Type

Driver

Modela el costo de procesamiento que requiere a un procesador la transferencia de un mensaje a través de un network. El driver modela el tiempo de overhead que supone para el procesador que envía o recibe mensajes, la ejecución de procesos de background que supervisan y gestionan el acceso al network.

Timer

Representa el overhead que introduce el timer del sistema sobre el procesador como consecuencia de las tareas de background que atienden los eventos que genera a nivel de interrupción.

Los objetos Timer modelan dos aspectos importantes del comportamiento de tiempo real:

- Los tiempos de overhead que afectan al procesador.
- La granularidad en la medida de tiempo que introduce y que se traduce en jitter en la ejecución de las operaciones.

Private Attributes:

Worst_Overhead : Normalized_Execution_Time = 0.0

Máximo tiempo (peor caso) de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.

Avg_Overhead : Normalized_Execution_Time = 0.0

Tiempo promedio de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.

Best_Overhead : Normalized_Execution_Time = 0.0

Mínimo tiempo (mejor caso) de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.

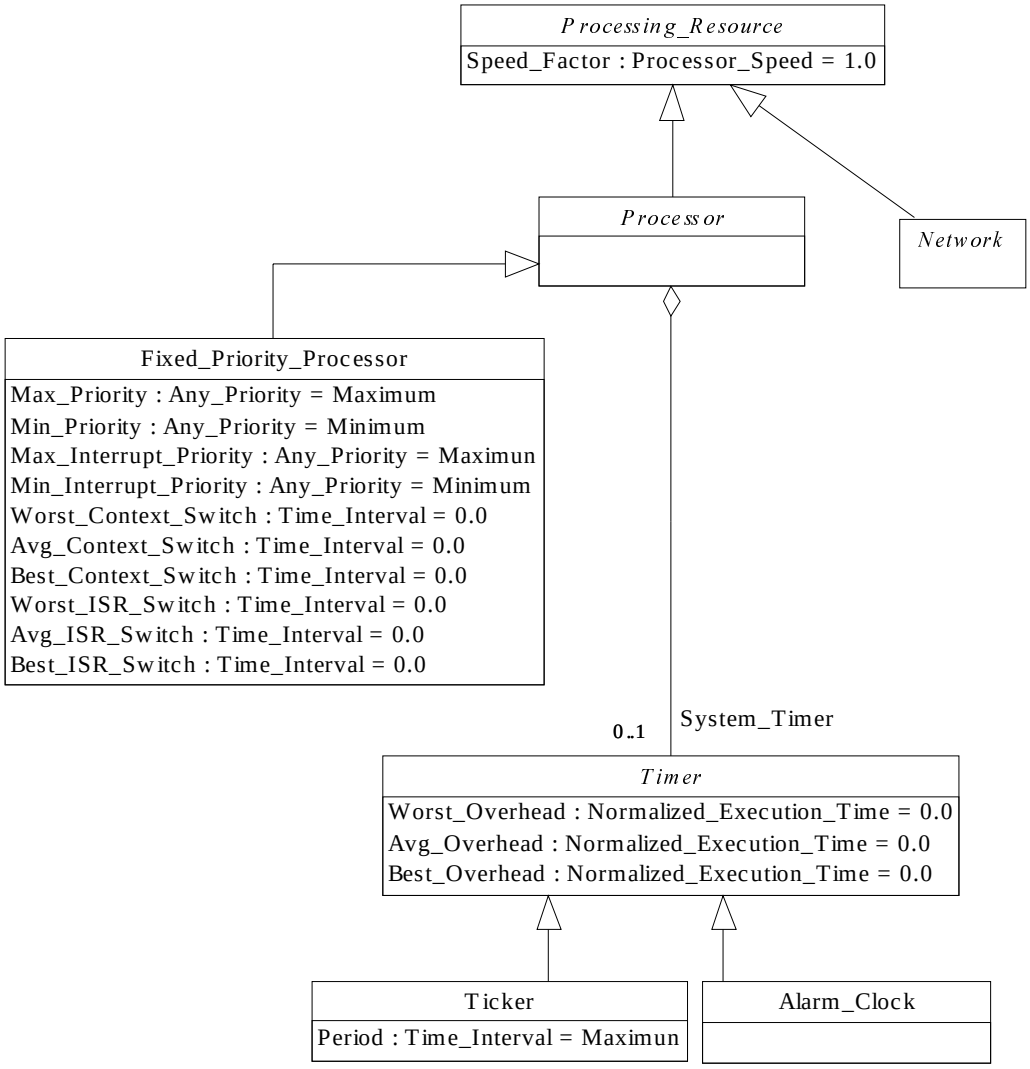


Diagrama 4.- Processor classes.

Processing_Resource

Modela un componente que ejecuta actividades que son parte del sistema que se modela.

Modela los componentes hardware y el software de background (sistema operativo, drivers, etc) en los que se ejecutan las actividades del sistema.

Private Attributes:

Speed_Factor : Processor_Speed= 1.0

Factor de velocidad de procesamiento del recurso. El tiempo real de ejecución de cualquier actividad que se ejecute en el recurso se obtendrá dividiendo el tiempo normalizado de ejecución (Worst_Case_Execution_Time,Avg_Case_Execution_Time, Best_Case_Execution_Time, etc.) establecido en las operaciones que lleva a cabo por el factor de velocidad.

Processor

Recurso de procesamiento que ejecuta código de la aplicación. Su capacidad de cómputo se distribuye entre la ejecución de código de las actividades del sistema que tiene asignadas y la ejecución de las tareas de gestión, monitorización y cambio de contexto entre actividades.

Derived from: Processing_Resource

Fixed_Priority_Processor

Procesador especializado que opera bajo una estrategia de prioridad fija.

Derived from: Processor

Private Attributes:

Max_Priority : Any_Priority = Maximum

Máximo nivel de prioridad para una actividad de procesado que se ejecuta en el procesador.

Min_Priority : Any_Priority = Minimum

Mínimo nivel de prioridad para una actividad de procesado que se ejecuta en el procesador.

Max_Interrupt_Priority : Any_Priority = Maximun

Máximo nivel de prioridad para una rutina de interrupción que se ejecuta en el procesador.

Min_Interrupt_Priority : Any_Priority = Minimum

Mínimo nivel de prioridad para una rutina de interrupción que se ejecuta en el procesador.

Worst_Context_Switch : Time_Interval = 0.0

Tiempo máximo de cómputo (peor caso) que el procesador emplea en realizar un cambio de contexto entre dos actividades de diferentes scheduling server.

Avg_Context_Switch : Time_Interval = 0.0

Tiempo promedio de cómputo que el procesador emplea en realizar un cambio de contexto entre dos actividades de diferentes scheduling server.

Best_Context_Switch : Time_Interval = 0.0

Tiempo mínimo de cómputo (mejor caso) que el procesador emplea en realizar un cambio de contexto entre dos actividades de diferentes scheduling server.

Worst_ISR_Switch : Time_Interval = 0.0

Tiempo máximo de cómputo (peor caso) que el procesador emplea en conmutar a una rutina de interrupción.

Avg_ISR_Switch : Time_Interval = 0.0

Tiempo promedio de cómputo que el procesador emplea en conmutar a la rutina de atención de una interrupción.

Best_ISR_Switch : Time_Interval = 0.0
Tiempo mínimo de cómputo (mejor caso) que el procesador emplea en conmutar a una rutina de interrupción.

Timer

Representa el overhead que introduce el timer del sistema sobre el procesador como consecuencia de las tareas de background con que se atienden los eventos que genera a nivel de interrupción.

Los objetos Timer modelan dos aspectos importantes del comportamiento de tiempo real:

- Los tiempos de overhead que afectan al procesador.
- La granularidad en la medida de tiempo que introduce y que se traduce en jitter de la ejecución de las operaciones.

Private Attributes:
Worst_Overhead : Normalized_Execution_Time = 0.0
Máximo tiempo (peor caso) de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.
Avg_Overhead : Normalized_Execution_Time = 0.0
Tiempo promedio de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.
Best_Overhead : Normalized_Execution_Time = 0.0
Mínimo tiempo (mejor caso) de cómputo que el procesador emplea en realizar una evaluación de los procesos en espera de temporización.

Ticker

Representa un timer basado en interrupciones hardware periódicas. En cada interrupción del temporizador el procesador evalúa si se ha alcanzado alguno de los plazos de temporización pendientes, y en caso de que se haya alcanzado se gestiona su atención.
Derived from: Timer

Private Attributes:
Period : Time_Interval = Maximun
Período de tiempo entre dos evaluaciones del estado del timer. Este tiempo representa la granularidad de apreciación de tiempo del procesador.

Alarm_Clock

Representa un temporizador basado en un hardware programable que interrumpe cuando se alcanza el plazo programado.
Derived from: Timer

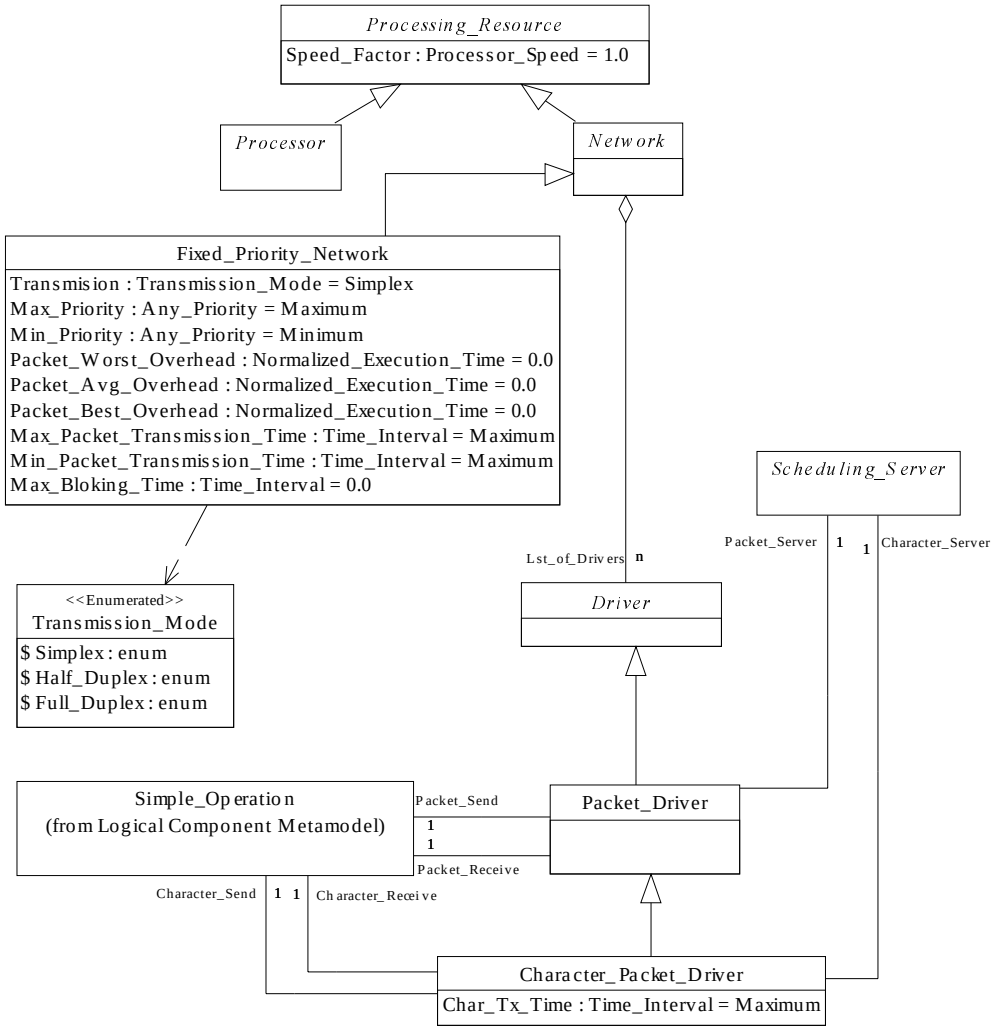


Diagrama 5.- Network classes.

Processing_Resource

Modela un componente que ejecuta actividades que son parte del sistema que se modela.

Modela los componentes hardware y el software de background (sistema operativo, drivers, etc) en los que se ejecutan las actividades del sistema.

Private Attributes:

Speed_Factor : Processor_Speed=1.0

Factor de velocidad de procesamiento del recurso. El tiempo real de ejecución de cualquier actividad que se ejecute en el recurso se obtendrá dividiendo el tiempo normalizado de ejecución (Worst_Case_Execution_Time,Avg_Case_Execution_Time, Best_Case_Execution_Time, etc.) establecido en las operaciones que lleva a cabo por el factor de velocidad.

Network

Recurso de procesamiento especializado que modela la ejecución de las operaciones de transferencia de mensajes entre procesadores a través de un mecanismo de comunicación existente en la plataforma. Modela el tiempo finito que requiere transferir un mensaje entre dos procesadores y la contención que produce en los procesos que realizan la transferencia el hecho de que estos deben transferirse uno a uno con exclusión mutua.

Derived from: Processing_Resource

Fixed_Priority_Network

Network especializado que realiza la comunicación, descomponiendo los mensajes en paquetes de longitud acotada que son transferidos bajo una política basada en prioridades estáticas. En cada instante solo se puede estar transfiriendo un paquete por el Network. La transmisión de un paquete es una actividad no interrumpible. La transmisión de un paquete requiere la dedicación del recurso durante un tiempo en el rango (Min_Packet_Transmission_Time .. Max_Packet_Transmission_Time). Así mismo, la transmisión de un paquete consume un overhead del propio Network (Packet_Overhead) como consecuencia de su uso en los protocolos de arbitraje de las prioridades de transmisión.

Derived from: Network

Private Attributes:

Transmision : Transmission_Mode = Simplex
Modo de transmisión (Simplex, Half_Duplex, Full_Duplex) en que opera el network.
Max_Priority : Any_Priority = Maximum
Máximo nivel de prioridad que puede tener un mensaje que se transfiere por el network.
Min_Priority : Any_Priority = Minimum
Mínimo nivel de prioridad que puede tener un mensaje que se transfiere por el network.
Packet_Worst_Overhead : Normalized_Execution_Time = 0.0
Tiempo máximo de ocupación del network (peor caso) que requiere el protocolo asociado a la transmisión de un paquete.
Packet_Avg_Overhead : Normalized_Execution_Time = 0.0

Tiempo promedio de ocupación del network que requiere el protocolo asociado a la transmisión de un paquete.

Packet_Best_Overhead : Normalized_Execution_Time = 0.0

Tiempo mínimo de ocupación del network (peor caso) que requiere el protocolo asociado a la transmisión de un paquete.

Max_Packet_Transmission_Time : Time_Interval = Maximum

Máximo tiempo (peor caso) que se utiliza del network para transmitir físicamente un paquete.

Min_Packet_Transmission_Time : Time_Interval = Maximum

Mínimo tiempo (mejor caso) que se utiliza del network para transmitir físicamente un paquete.

Max_Bloking_Time : Time_Interval = 0.0

Tiempo máximo que permanece ocupado el network como consecuencia de la transferencia de un paquete. Este tiempo incluye tanto la transferencia del paquete como el tiempo debido a cabeceras de control del paquete o a protocolos de determinación del paquete de mayor prioridad.

Transmission_Mode

Modos de transmisión en que puede operar un Network.

Private Attributes:

Simplex : enum

El Network sólo admite transmisión de mensajes en un sentido.

Half_Duplex : enum

El Network permite la transmisión de información en los dos sentidos, pero son mutuamente exclusivos en el tiempo.

Full_Duplex : enum

El Network permite la transmisión de información en los dos sentidos y ambos procesos pueden ser coincidentes en el tiempo

Driver

Modela el costo de procesamiento que requiere a un procesador la transferencia de un mensaje a través de un network. El driver modela el tiempo de overhead que supone para el procesador que envía o recibe mensajes la ejecución de procesos de background que supervisan y gestionan el acceso al network.

Packet_Driver

Driver especializado basado en un único Scheduling_Server (Driver_ Scheduling_Server)que soporta una tarea (de Background) que es planificada dentro del Processor donde está instalado el driver. Por cada paquete que recibe o que envía, ejecuta la correspondiente operación (Packet_Send o Packet_Receive) que tiene asociada.

Derived from: Driver

Character_Packet_Driver

Packet driver especializado que requiere asistencia del procesador para el envío de cada uno de los caracteres que componen el paquete.

Derived from: Packet_Driver

Private Attributes:
Char_Tx_Time : Time_Interval = Maximum
Tiempo que esta ocupado el network cuando transfiere un carácter.

Scheduling_Server

Modela un proceso activo simple (con un solo thread) dentro del que se ejecutan una o varias actividades secuencialmente y que es ejecutado por el recurso de procesamiento que tiene referenciado.

Así mismo, modela la política de planificación y la capacidad de procesamiento disponible para ejecutar las actividades que se le asignan.

Derived from: Job_Parameter_Type

Simple_Operation (from Logical Component Metamodel)

Descripción de la temporización de una operación simple. La operación que se modela se ejecuta de forma autónoma salvo en que compite con otras operaciones concurrentes en el uso de recursos compartidos o de recursos de procesado.

Derived from: Operation

Private Attributes:
WCET : Normalized_Execution_Time = 0.0
Tiempo máximo (peor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
ACET : Normalized_Execution_Time = 0.0
Tiempo promedio que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
BCET : Normalized_Execution_Time = 0.0
Tiempo mínimo (mejor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.

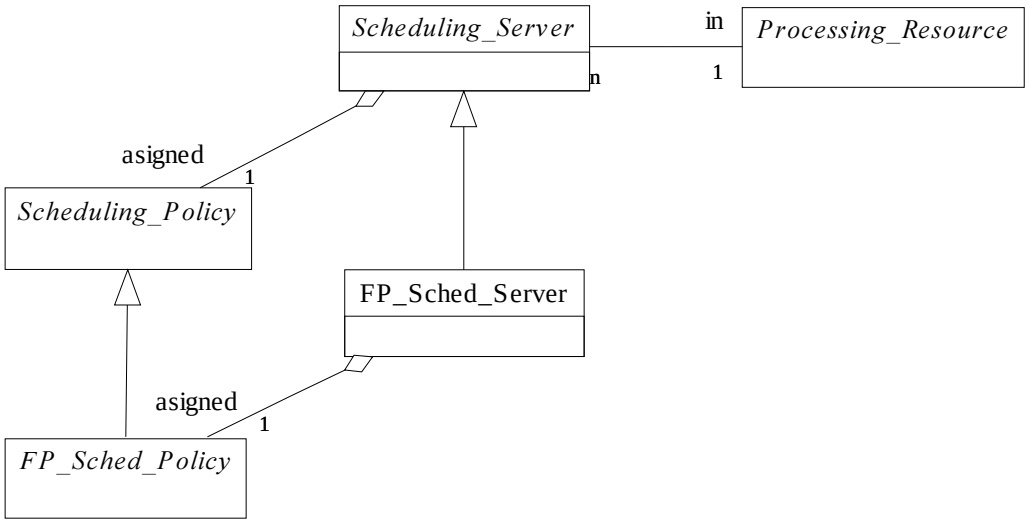


Diagrama 6.- Scheduler Server classes.

Scheduling_Server

Modela un proceso activo simple (con un solo thread) dentro del que se ejecutan una o varias actividades secuencialmente y que es ejecutado por el recurso de procesamiento que tiene referenciado.

Así mismo, modela la política de planificación y la capacidad de procesamiento disponible para ejecutar las actividades que se le asignan.

Derived from: Job_Parameter_Type

Scheduling_Policy

Estrategia de planificación que se utiliza para planificar las actividades que se ejecutan dentro de un scheduling_server.

Processing_Resource

Modela un componente que ejecuta actividades que son parte del sistema que se modela.

Modela los componentes hardware y el software de background (sistema operativo, drivers, etc) en los que se ejecutan las actividades del sistema.

FP_Sched_Server

Sistema de planificación que ejecuta las actividades que tiene asociadas con exclusión mútua entre ellas, con una política de planificación basada en prioridades fijas.

Derived from: Scheduling_Server

FP_Sched_Policy

Política de planificación compatible con un FP_Sched_Server.

Derived from: Scheduling_Policy

Private Attributes:

The_Priority : Any_Priority = Minimum

Nivel de prioridad asignada al servidor.

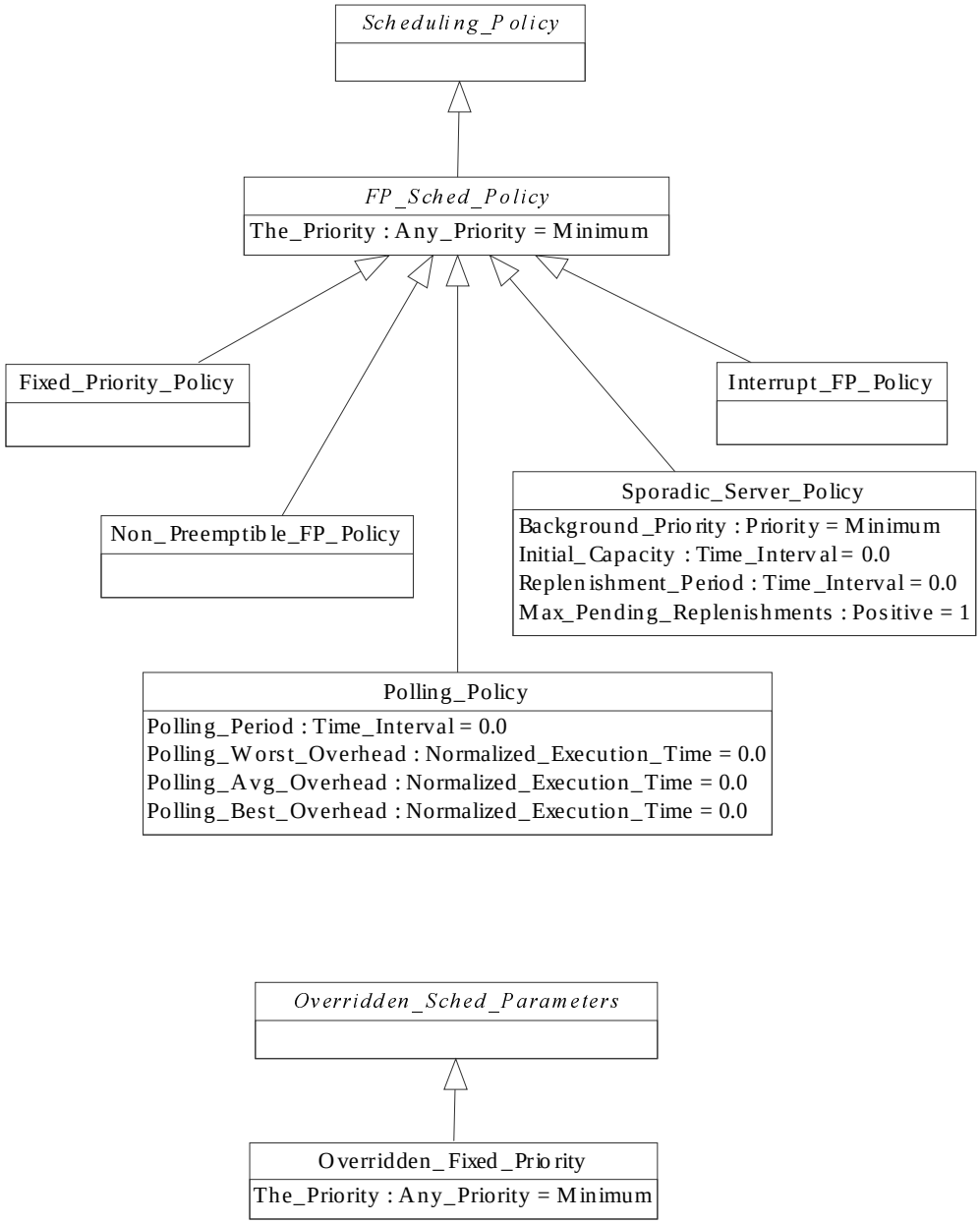


Diagrama 7.- Scheduling Policy classes.

Scheduling_Policy

Estrategia de planificación que se utiliza para planificar las actividades que se ejecutan dentro de un scheduling_server.

FP_Sched_Policy

Política de planificación compatible con un FP_Sched_Server.

Derived from: Scheduling_Policy

Private Attributes:
The_Priority : Any_Priority = Minimum
Nivel de prioridad asignada al servidor.

Fixed_Priority_Policy

Planificación de prioridad fija con expulsión.

Derived from: FP_Sched_Policy

Non_Preemptible_FP_Policy

Planificación de prioridad fija sin expulsión.

Derived from: FP_Sched_Policy

Polling_Policy

Política de planificación basada en prioridades fijas y en el escrutinio periódico de la tabla de procesos dispuestos para su planificación.

Derived from: FP_Sched_Policy

Public Attributes:
Polling_Period : Time_Interval = 0.0
Periodo de escrutinio del planificador.
Polling_Worst_Overhead : Normalized_Execution_Time = 0.0
Tiempo normalizado máximo (peor caso) de sobrecarga debida a la planificación.
Polling_Avg_Overhead : Normalized_Execution_Time = 0.0
Tiempo normalizado promedio de sobrecarga debida a la planificación.
Polling_Best_Overhead : Normalized_Execution_Time = 0.0
Tiempo normalizado mínimo (mejor caso) de sobrecarga debida a la planificación.

Sporadic_Server_Policy

Planificador basado en un servidor esporádico.

Nota: En este caso The_Priority representa la prioridad normal (alta) con la que se planifican las actividades cuando el servidor aun dispone de parte de la capacidad de ejecución que tiene asignada.

Derived from: FP_Sched_Policy

Public Attributes:

Background_Priority : Priority = Minimum

Prioridad reducida de fondo (baja) a la que procesa los eventos que llegan una vez que haya completado la capacidad de ejecución que tiene asignada.

Initial_Capacity : Time_Interval = 0.0

Tiempo de ejecución de que inicialmente dispone el servidor esporádico para atender la actividades que tiene asociadas.

Replenishment_Period : Time_Interval = 0.0

Periodo de relleno durante el que procesa los eventos al nivel de prioridad de fondo, una vez que se haya completado la capacidad de ejecución que tiene asignada.

Max_Pending_Replenishments : Positive = 1

Número máximo de eventos pendientes que puede tener suspendidos el servidor para ser posteriormente atendidos

Interrupt_FP_Policy

Política de planificación expulsable pero con prioridades definidas en el rango de prioridades de interrupción.

Derived from: FP_Sched_Policy

Overridden_Sched_Parameters

Subconjunto de parámetros de planificación que las operaciones pueden imponer al servidor bajo el que se ejecuta.

Este componente sólo se incluye cuando en el propio código de la operación hay sentencias que modifican explícitamente la prioridad del thread en que se ejecuta.

Overridden_Fixed_Priority

Subconjunto de parámetros de planificación correspondientes a un planificador de prioridad fija y con expulsión, que una operación puede imponer durante su ejecución al servidor de planificación en que se ejecuta.

Derived from: Overridden_Sched_Parameters

Public Attributes:

The_Priority : Any_Priority = Minimum

Nivel de prioridad que se impone durante la ejecución de la operación.

Logical Component Metamodel

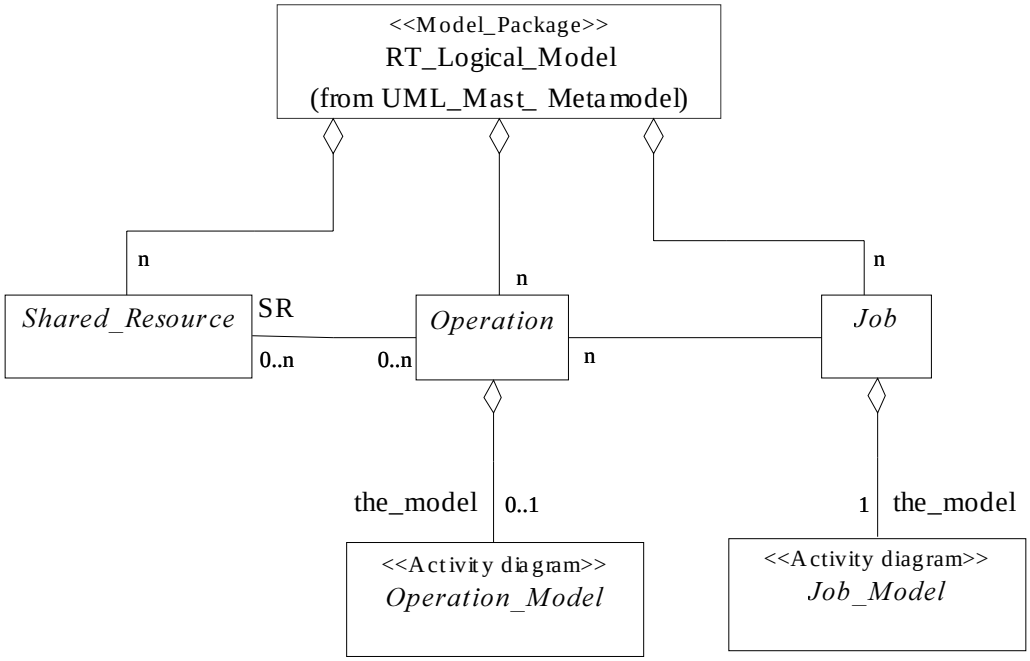


Diagrama 8.- RT_Logical_Model.

RT_Logical_Model (From UML-Mast Metamodel)

Modela los requerimientos de procesado que requiere la ejecución de las operaciones funcionales definidas en los componentes lógicos que se utilizan en el diseño, tales como métodos, procedimientos y funciones definidos en las clases, primitivas de sincronización entre threads, sesiones de comunicación, etc.

Describe el comportamiento de tiempo real de los componentes funcionales (clases, métodos, procedimientos, operaciones, etc.) que están definidos en el sistema y cuyos tiempos de ejecución condicionan el cumplimiento de sus especificaciones del tiempo real.

El RT_Logical_Model modela dos aspectos de un componente:

- La complejidad de los algoritmos con que se implementa.
- Las posibilidades de bloqueo que pueden retrasar su ejecución y que son consecuencia de tener que hacer uso exclusivo de recursos compartidos (Shared_Resource).

Shared_Resource

Modela un recurso que es requerido en régimen de exclusión mutua por diferentes operaciones concurrentes y es causa potencial de posibles bloqueos al acceder a él.

Derived from: Oper_Parameter_Type

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

Operation_Model

Diagrama de actividad que describe la secuencia de operaciones más simples de que se compone la operación.

Contiene la secuencia ordenada de operaciones primitivas que implica su ejecución. El modelo de una operación se describe mediante un diagrama de actividad UML agregado a la operación y consistente en una única activity state de UML con una sentencia do/ por cada operación primitiva de que se compone.

Derived from: Activity_Model

Job

Modela la actividad potencialmente concurrente que se desencadena cuando se ejecuta un componente lógico (procedimiento, función método, etc.) en un programa concurrente. La actividad modelada por un job puede requerir que intervengan múltiples threads y puede perdurar aún después de que haya concluido la invocación del componente que lo desencadenó.

Derived from: Job_Parameter_Type

Job_Model

Diagrama de actividad que describe las actividades, acciones y scheduling_server, de que se compone un job.

El Job_Model es un Activity_Model ampliado con Conector_State.

Derived from: Activity_Model

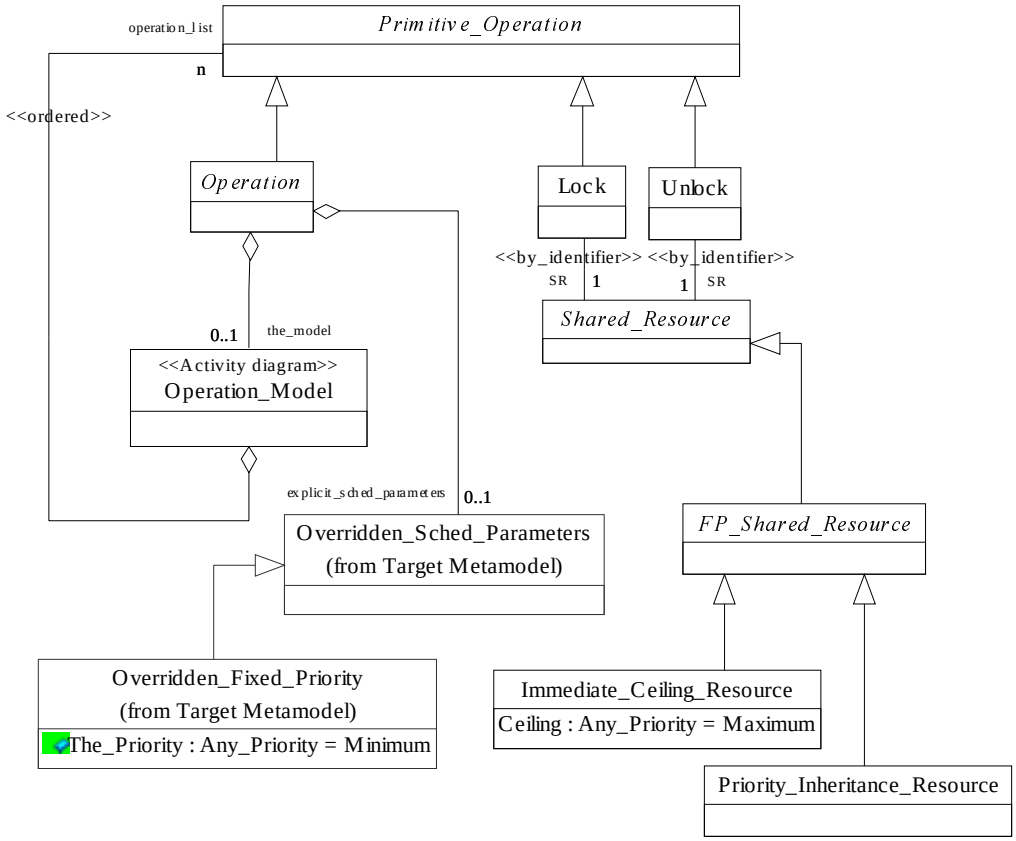


Diagrama 9.- Primitive Operation and Shared Resource classes.

Primitive_Operation

Representa cualquier tipo de operación que puede declarase. Es una clase abstracta que se introduce para definir la operaciones elementales de las que puede componerse una operación compuesta.

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

Operation_Model

Diagrama de actividad que describe la secuencia de operaciones más simples de que se compone la operación.

Contiene la secuencia ordenada de operaciones primitivas que implica su ejecución. El modelo de una operación se describe mediante un diagrama de actividad UML agregado a la operación y consistente en una única actividad UML con una sentencia do/ por cada operación primitiva de que se compone.

Derived from: Activity_Model

Overridden_Sched_Parameters (From Target Metamodel)

Subconjunto de parámetros de planificación que las operaciones pueden imponer al servidor bajo el que se ejecuta.

Este componente sólo se incluye cuando en el propio código de la operación hay sentencias que modifican explícitamente la prioridad del thread en que se ejecuta.

Derived from: Oper_Parameter_Type

Overridden_Fixed_Priority(From Target Metamodel)

Subconjunto de parámetros de planificación correspondientes a un planificador de prioridad fija y con expulsión, que una operación puede imponer durante su ejecución al servidor de planificación en que se ejecuta.
Derived from: Overriden_Sched_Parameters

Private Attributes:
The_Priority : Any_Priority = Minimum
Prioridad que se impone durante la ejecución de la operación.

Lock

Operación primitiva que espera a que un Shared_Resource esté libre y que termina cuando accede a él en régimen exclusivo.

Derived from: Primitive_Operation

Unlock

Operación primitiva que representa la liberación de un recurso compartido previamente tomado.

Derived from: Primitive_Operation

Shared_Resource

Modela un recurso que es requerido en régimen de exclusión mutua por diferentes operaciones concurrentes y es causa potencial de posibles bloqueos al acceder a él.

Derived from: Oper_Parameter_Type

FP_Shared_Resource

Recurso compartido al que se accede siguiendo un criterio de prioridad fija.

Derived from: Shared_Resource

Immediate_Ceiling_Resource

Recurso compartido al que se accede siguiendo un criterio de prioridad fija. Cuando una actividad lo tiene tomado es planificada de acuerdo al criterio de Techo de Prioridad, esto es, la actividad se ejecuta a una prioridad igual a la mayor de entre la de los proceso que pueden hacer uso del recurso. Esta prioridad se calcula y establece estáticamente.

Derived from: FP_Shared_Resource

Private Attributes:

Ceiling : Any_Priority = Maximum

Techo de prioridad del recurso: prioridad a la que se ejecutan los procesos durante el tiempo que tienen reservado el recurso

Priority_Inheritance_Resource

Recurso compartido al que se accede siguiendo un criterio de prioridad fija. Cuando una actividad lo tiene reservado es planificada de acuerdo al criterio de Herencia de Prioridad, esto es, durante ese tiempo la actividad es planificada con la prioridad de la tarea que tiene mayor prioridad de entre las que esperan el acceso al recurso.

Derived from: FP_Shared_Resource

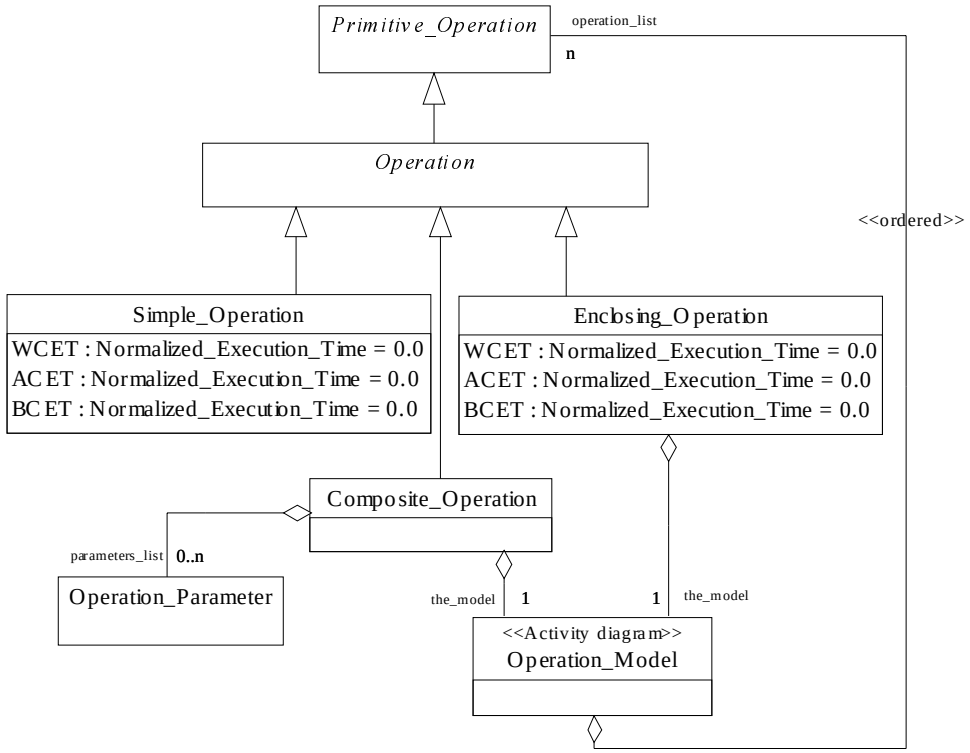


Diagrama 10.- Operation clases.

Primitive_Operation

Representa cualquier tipo de operación que puede declarase. Es una clase abstracta que se introduce para definir la operaciones elementales de las que puede componerse una operación compuesta.

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

Simple_Operation

Descripción de la temporización de una operación simple. La operación que se modela se ejecuta de forma autónoma salvo en que compite con otras operaciones concurrentes en el uso de recursos compartidos o de recursos de procesado.

Derived from: Operation

Private Attributes:
WCET : Normalized_Execution_Time = 0.0
Tiempo máximo (peor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
ACET : Normalized_Execution_Time = 0.0
Tiempo promedio que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
BCET : Normalized_Execution_Time = 0.0
Tiempo mínimo (mejor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.

Enclosing_Operation

Representa una operación compuesta en la que solo se enumeran algunas de las operaciones que lleva a cabo. Se considera implícitamente que además de las especificadas en la lista (Composite_Operations_List), existen otras operaciones simples con las siguientes restricciones:

- Las operaciones no incluidas explícitamente no requieren ni liberan recursos.

- La suma de los correspondientes tiempos de ejecución de todas las operaciones que constituyen la Enclosing_Operation, supuesto que no existen bloqueos por no disponibilidad de recursos, es la que define los valores de sus atributos (WCET, ACET y BCET).

Su descripción es similar a la de una Composite_Operation, salvo que no admite parámetros.

Derived from: Operation

Private Attributes:
WCET : Normalized_Execution_Time = 0.0
Tiempo mínimo (mejor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
ACET : Normalized_Execution_Time = 0.0
Tiempo promedio que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.
BCET : Normalized_Execution_Time = 0.0
Tiempo mínimo (mejor caso) que tarda en ejecutarse la operación. Se expresa en unidades de tiempo relativas al speed_factor del procesador en que se ejecuta.

Composite_Operation

Operación lineal que describe una secuencia ordenada de operaciones.

Cada ejecución de una operación compuesta se ejecuta siempre dentro de un mismo servidor de planificación, sin embargo los parámetros de planificación pueden cambiar durante su ejecución si las operaciones simples que la componen tienen asociados parámetros de planificación específicos.

Derived from: Operation

Operation_Model

Diagrama de actividad que describe la secuencia de operaciones más simples de que se compone la operación.

Contiene la secuencia ordenada de operaciones primitivas que implica su ejecución. El modelo de una operación se describe mediante un diagrama de actividad UML agregado a la operación y consistente en una única activity state de UML con una sentencia do/ por cada operación primitiva de que se compone.

Derived from: Activity_Model

Operation_Parameter

Parámetro declarado en una operación compuesta.

Un parámetro de una operación compuesta se describe como un atributo en la clase con la que se declara la operación compuesta.

Private Attributes:

Name : Identifier

Identificador del parámetro. Cada parámetro de una operación compuesta tiene un identificador único dentro de la operación compuesta a la que pertenece.
El parámetro de una operación compuesta se describe en el modelo como un atributo de la clase que la declara. El identificador del parámetro se describe como el identificador del atributo.

Value : Oper_Parameter_Type

El valor de un parámetro es el valor que se asigna al parámetro en la invocación de la operación compuesta dentro de su descripción. Esto se hace mediante una sentencia do/ con el formato:

```
do/ The_Composite_Operation (Param_Ident_1 => Param_Value_1,
    Param_Ident_2 => Param_Value_2, .....)
```

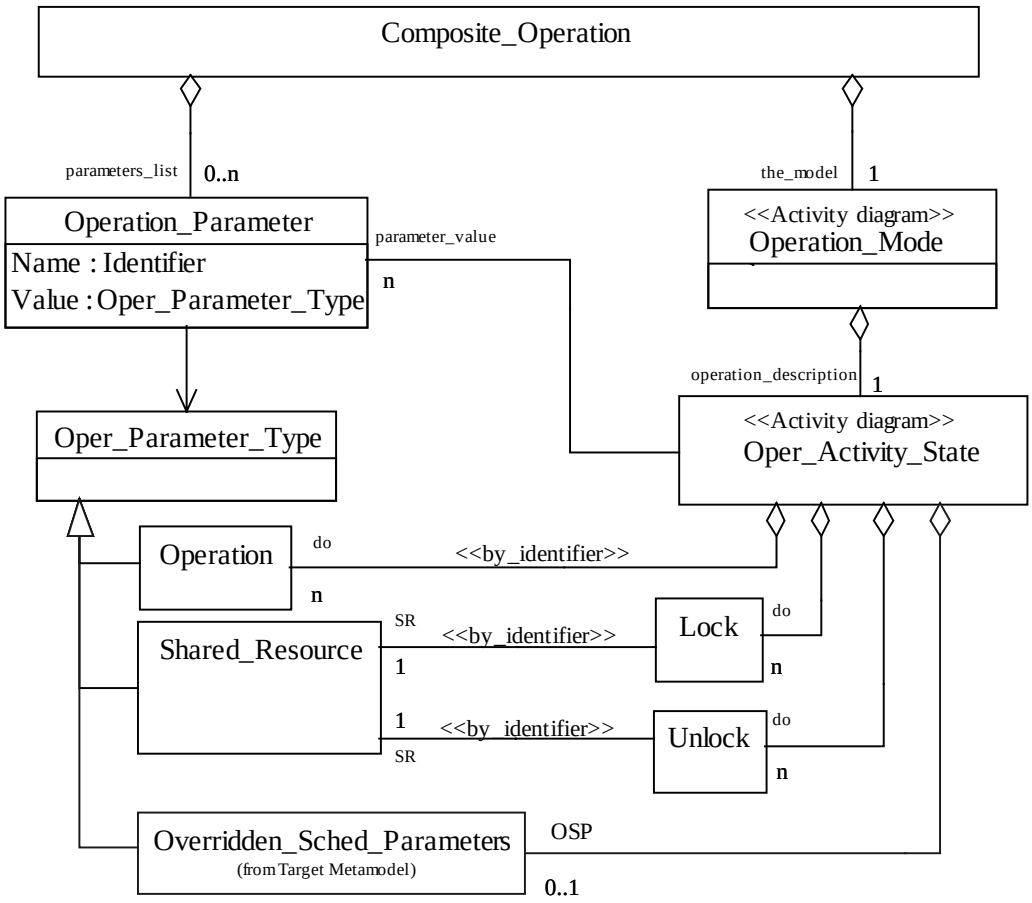


Diagrama 11: Composite Operation parameter classes.

Composite_Operation

Operación lineal que describe una secuencia ordenada de operaciones.

Cada ejecución de una operación compuesta se ejecuta siempre dentro de un mismo servidor de planificación, sin embargo los parámetros de planificación pueden cambiar durante su ejecución si las operaciones simples que la componen tienen asociados parámetros de planificación específicos.

Derived from: Operation

Operation_Parameter

Parámetro declarado en una operación compuesta.

Un parámetro de una operación compuesta se describe como un atributo en la clase con la que se declara la operación compuesta.

Private Attributes:

Name : Identifier

Identificador del parámetro. Cada parámetro de una operación compuesta tiene un identificador único dentro de la operación compuesta a la que pertenece.

El parámetro de una operación compuesta se describe en el modelo como un atributo de la clase que la declara. El identificador del parámetro se describe como el identificador del atributo.

Value : Oper_Parameter_Type

El valor de un parámetro es el valor que se asigna al parámetro en la invocación de la operación compuesta dentro de su descripción. Esto se hace mediante una sentencia do/ con el formato:

```
do/ The_Composite_Operation (Param_Ident_1 => Param_Value_1,
    Param_Ident_2 => Param_Value_2, .....)
```

Oper_Parameter_Type

Tipo del parámetro de operación. Los parámetros de las operaciones compuestas deben ser de uno de los tres tipos siguientes:

- Operation
- Shared_Resource
- Overridden_Sched_Parameter

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

Shared_Resource

Modela un recurso que es requerido en régimen de exclusión mutua por diferentes operaciones concurrentes y es causa potencial de posibles bloqueos al acceder a él.

Derived from: Oper_Parameter_Type

Overridden_Sched_Parameters

Subconjunto de parámetros de planificación que las operaciones pueden imponer al servidor bajo el que se ejecuta.

Este componente es optativo y sólo debe incluirse cuando en el propio código de la operación hay sentencias que modifican explícitamente la prioridad del thread en que se ejecuta.

Derived from: **Oper_Parameter_Type**
Job_Parameter_Type

Operation_Model

Diagrama de actividad que describe la secuencia de operaciones más simples de que se compone la operación.

Contiene la secuencia ordenada de operaciones primitivas que implica su ejecución. El modelo de una operación se describe mediante un diagrama de actividad UML agregado a la operación y consistente en una única activity state UML con una sentencia do/ por cada operación primitiva de que se compone.

Derived from Activity_Model

Oper_Activity_State

Describe una Composite_Operation y es un activity state tal como se define en UML que tiene declarada una lista ordenada de sentencias do/ de uno de los siguientes tipos:

```
do/ Operation
do/ Lock(SR=> Shared_Resource)
do/ Unlock(SR=>Shared_Resource)
do/ Composite_Operation_x (OSP=> Overriden_Sched_Parameter, … )
do/ Composite_Operation_y(Oper_Parameter_name_1 =>
                                    Oper_Parameter_Type_Identifier_1, … )
```

Lock

Operación primitiva que espera a que un Shared_Resource esté libre y que termina cuando accede a él en régimen exclusivo.

Derived from:Primitive_Operation

Unlock

Operación primitiva que representa la liberación de un recurso compartido previamente tomado.

Derived from: Primitive_Operation

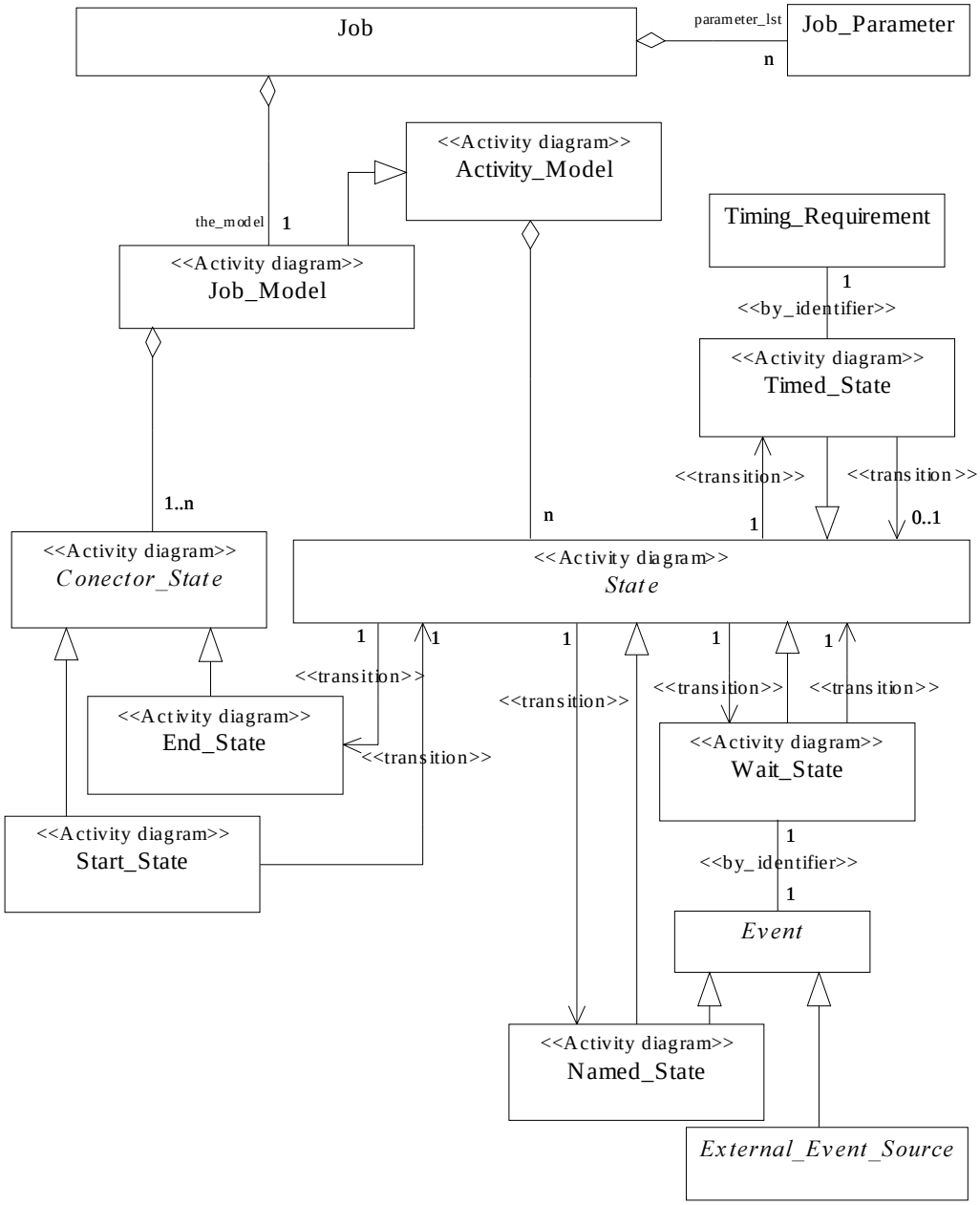


Diagrama 12.- Job classes.

Job

Modela la actividad potencialmente concurrente que se desencadena cuando se ejecuta un componente lógico (procedimiento, función método, etc.) en un programa concurrente. La actividad modelada por un job puede requerir que intervengan múltiples threads y puede perdurar aún después de que haya concluido la invocación del componente que la desencadenó.

Derived from: Job_Parameter_Type

Job_Parameter

Los parámetros que se definen en un Job se declaran como atributos de la clase que define el Job. Los parámetros se declaran con su identificador, su tipo y, en su caso, el valor por defecto.

Private Attributes:

Name : Identifier

Nombre del parámetro del Job. En el modelo se representa por el identificador del atributo con que se declara el parámetro.

Value : Job_Parameter_Type

Valor que se asigna al parámetro en una invocación del job. El valor se asigna con una sentencia del tipo genérico:

do/ Job_name(Param_Ident=> Param_Value)

Activity_Model

Describe el modelo de actividad (Activity_Model) de una Transaction o Job.

El modelo de actividad está compuesto de un conjunto de diagramas de actividad que describen las secuencias de actividades, estados y transiciones que se desencadenan como consecuencia de un patrón de eventos externos de entrada

Job_Model

Diagrama de actividad que describe las actividades, acciones y scheduling server de que se compone un job.

El Job_Model es un Activity_Model ampliado con Conector_State.

Derived from: Activity_Model

Conector_State

Son estados falsos o pseudoestados del diagrama de actividad que se introducen como elementos de interconexión entre módulos y son solo utilizados en los Job_Model. Se utilizan para representar:

- La transferencia del flujo de control desde el estado en que se invoca un job al estado en que se inicia el job invocado.
- La transferencia de flujo de un estado interno del job al estado que sigue a aquel desde el que se invocó.

Start_State

Representa el estado de partida en cada invocación del job. En cada Job_Model solo puede haber un Start_State. En el diagrama se representa mediante un pequeño circulo negro, símbolo UML de un Initial_State.

Derived from: Conector_State

End_State

Representa el estado de conclusión del procedimiento cuya invocación inició el job.

En cada job hay un solo End_State y se representa mediante un punto negro al interior de una circunferencia u ojo de buey, que es el símbolo UML de un Final_State.

Alcanzar el End_State no representa el fin del job en el sentido de que han terminado todas las actividades concurrentes que ha generado, sino que el job ha alcanzado un estado en el que retorna el flujo de control a la actividad que sigue a la invocación del job.

Derived from: Conector_State

State

Son las activity states y action states UML del diagrama de actividad que representan los estados que alcanza el job en su ejecución.

Timed_State

Representa un action state que se declara para asignar un requerimiento temporal (Timing requirement) al estado que representa dentro del job.

El nombre del Timed_State debe tener un identificador idéntico al del Timing_Requirement que se le asocia, el cual debe estar declarado en la transacción a la que corresponden ambos.

Un Timed_Sate tiene siempre una transición de entrada y puede tener una o ninguna transición de salida.

Derived from: State

Timing_Requirement

Representa el requerimiento temporal que se asocia a un estado (State_Action) dentro del contexto de una transacción.
Cada requerimiento temporal se define como relativo a un evento externo o a una transición que se encuentra dentro de la línea directa de flujo que ha provocado la transición a la que se asocia el requerimiento temporal.
Cuando una transición puede resultar como unión (join) de diferentes líneas de flujo de control con origen en diferentes fuentes de eventos externos, la restricción temporal asociada a ella solo se aplica a los eventos originados por la fuente de eventos externos a la que hace referencia el requerimiento temporal.
Los diferentes requerimientos temporales de un requerimiento compuesto se aplican con criterio conjuntivo (and) si son relativos a un mismo evento y con carácter disyuntivo (or) si es relativo a diferentes eventos.

Derived from: Job_Parameter_Type

Named_State

Representa un action state (y por tanto de duración nula) del Job_Model, al que se le asigna un nombre para poder referenciarlo dentro de la misma transacción. Un Named_State se utiliza en el modelo de actividad en los siguientes dos casos:

- Cuando se necesita hacer referencia al estado que representa fuera del modelo del Job. Esto ocurre, cuando se necesita que una de las líneas concurrentes de flujo de control del Job continúe en otra sección del modelo de la misma transacción.

- Para finalizar una línea de flujo concurrente que haya permanecido activa después de que haya finalizado la línea que corresponde a la invocación del Job.

El ámbito del identificador de un Named_State es la instancia del Job en que está declarado. Los Named_States que están declarado en un Job, se instancian como Named_State diferentes en cada instanciación del Job. Si diferentes Job necesitan hacer referencia a un mismo Name_State, este debe ser transferido a través de un parámetro. Un Named State se declara tanto en un Named_State del diagrama de actividad, como en un Wait_State que hace referencia al Named_State.

Un Name_State tiene siempre una transición de entrada y puede tener o no una transición de salida.

Derived from: **State,**
Event

Wait_State

Es un activity state en el que al ser alcanzado se suspende la correspondiente línea de flujo de control, a la espera de que se genere un evento externo o de que se alcance un Named_State en alguna sección del modelo de la transacción.

Si el Wait_State representa una espera a un evento externo, el Identificador del Wait_State debe ser el mismo que el identificador del External_Event_Source que caracteriza la temporización de los eventos externos y que debe haber sido declarado en la transacción.

Si el Wait_State representa una espera a que el modelo alcance un determinado estado, el identificador del Wait_State debe coincidir con el del Named_State a cuya espera se suspende. La combinación Named_State y Wait_State con el mismo identificador es el mecanismo que se dispone para transferir una línea de flujo de control concurrente de una sección de la transacción a otra.

Se impone la restricción de que solo puede existir un Wait_State por cada Named_State definido en el modelo del job.

Derived from: State

Event

Es una clase abstracta que generaliza de forma compatible los eventos externos (definidos por External_Event_Source)y los eventos internos (Named_State).

External_Event_Source

Representa una secuencia (stream) de eventos que tienen su origen externo al código del sistema. Pueden ser originados por el entorno o por componentes hardware del sistema (timer, etc). Lo específico de su patrón de generación es que es autónomo y no depende del estado o de la evolución del sistema que se está modelando.

Derived from: **Event,**
Job_Parameter_Type

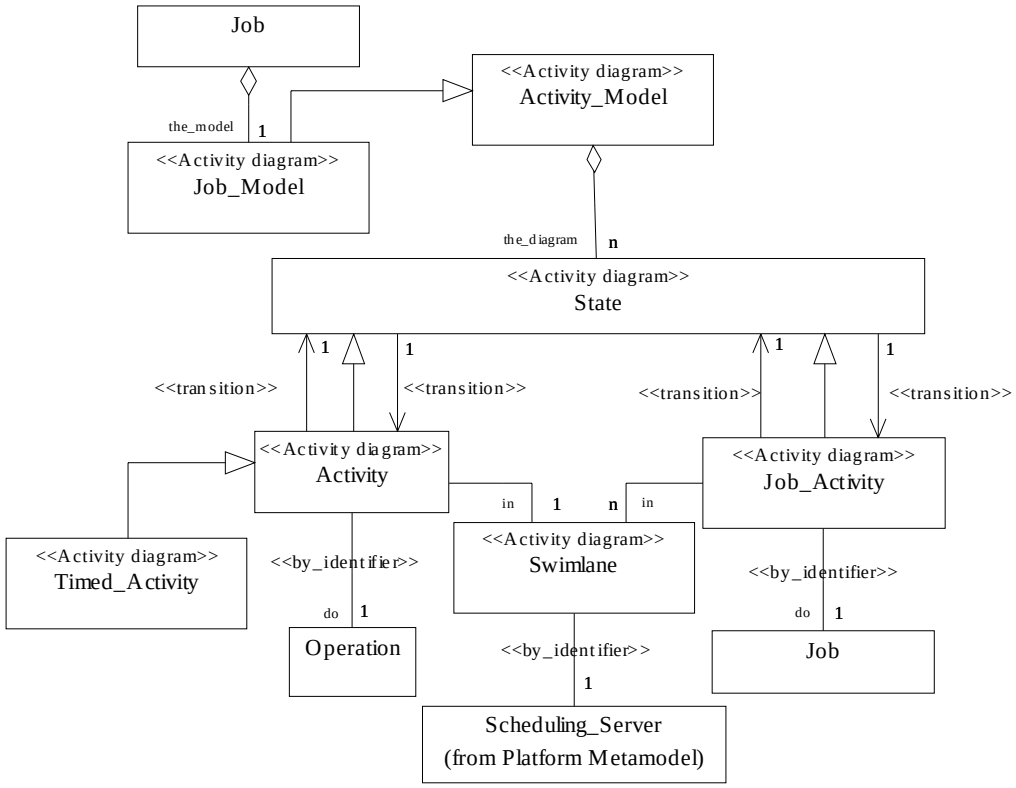


Diagrama 13.- Activities classes.

Job

Modela la actividad potencialmente concurrente que se desencadena cuando se ejecuta un componente lógico (procedimiento, función método, etc.) en un programa concurrente. La actividad modelada por un job puede requerir que intervengan múltiples threads y puede perdurar aún después de que haya concluido la invocación del componente que la desencadenó.

Derived from: Job_Parameter_Type

Activity_Model

Describe el modelo de actividad (Activity_Model) de una Transaction (o Job).

El modelo de actividad esta compuesto de un conjunto de diagramas de actividad que describen las secuencias de actividades, estados y transiciones que se desencadenan como consecuencia de un patrón de eventos externos de entrada.

Job_Model

Diagrama de actividad que describe las actividades, acciones y scheduling server, de que se compone un job.

El Job_Model es un Activity_Model ampliado con Conector_State.

Derived from: Activity_Model

State

Son las activity states y action states UML del diagrama de actividad que representan los estados que alcanza el job en su ejecución.

Activity

Es un Activity_State que ser alcanzado representa que se activa una nueva ejecución de una Operation que se declara en su sentencia do/. El estado se abandona cuando se termina la ejecución de la Operation, realizándose entonces, la transición de salida.

Sólo puede contener una sentencia do/.

Dado que en general los sistemas que se modelan son concurrentes, puede activarse una Activity antes que haya concluido su ejecución anterior. Cuando esto ocurre, habrá dos instancias de la Activity dispuestas para ser planificadas en el mismo Scheduling_Server. Ambas instancias comparten una misma política de planificación (y en su caso, tienen la misma prioridad).

La Operation que se referencia mediante su identificador en la sentencia do/ debe haber sido declarada y descrita en otra sección del modelo.

Las Activity de un Job_Model tienen una única transición de entrada y una única transición de salida.

Derived from: State

Timed_Activity

Representa una activity state que es activada por el timer del procesador. Si el timer es del tipo Alarm_Clock, existe una sección del código que se ejecuta a la prioridad de la interrupción del timer, que puede interrumpir actividades de mayor prioridad que ella. Si el timer es del tipo Ticker, solo se diferencia por la presencia de una granularidad extra en el tiempo de disparo y en que se analiza como un termino adicional de jitter.

Derived from: Activity

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

Swimlane

Banda vertical del diagrama de actividad que representa a cada Scheduling_Server que participa en la ejecución del job.

Cada Activity del Job debe situarse en la Swimlane que corresponde al Scheduling_Server al que se asigna su ejecución El Scheduling_Server en que se ejecutan las actividades declaradas en un Job se especifica en la propia descripción del Job.

Las Activity que en la descripción de un Job se asigna a un Swinlane sin nombre, se ejecutan en el Scheduling_Server en que se encuentra la Activity que invoca al Job.

Scheduling_Server

Modela un proceso activo simple (con un solo thread) dentro del que se ejecutan una o varias actividades secuencialmente y que es ejecutado por el recurso de procesamiento que tiene referenciado.

Así mismo, modela la política de planificación y la capacidad de procesamiento disponible para ejecutar las actividades que se le asignan.

Derived from: Job_Parameter_Type

Job_Activity

Es un Activity_State que representa la activación de una nueva ejecución del job que se declara en su sentencia do/.

Derived from: State

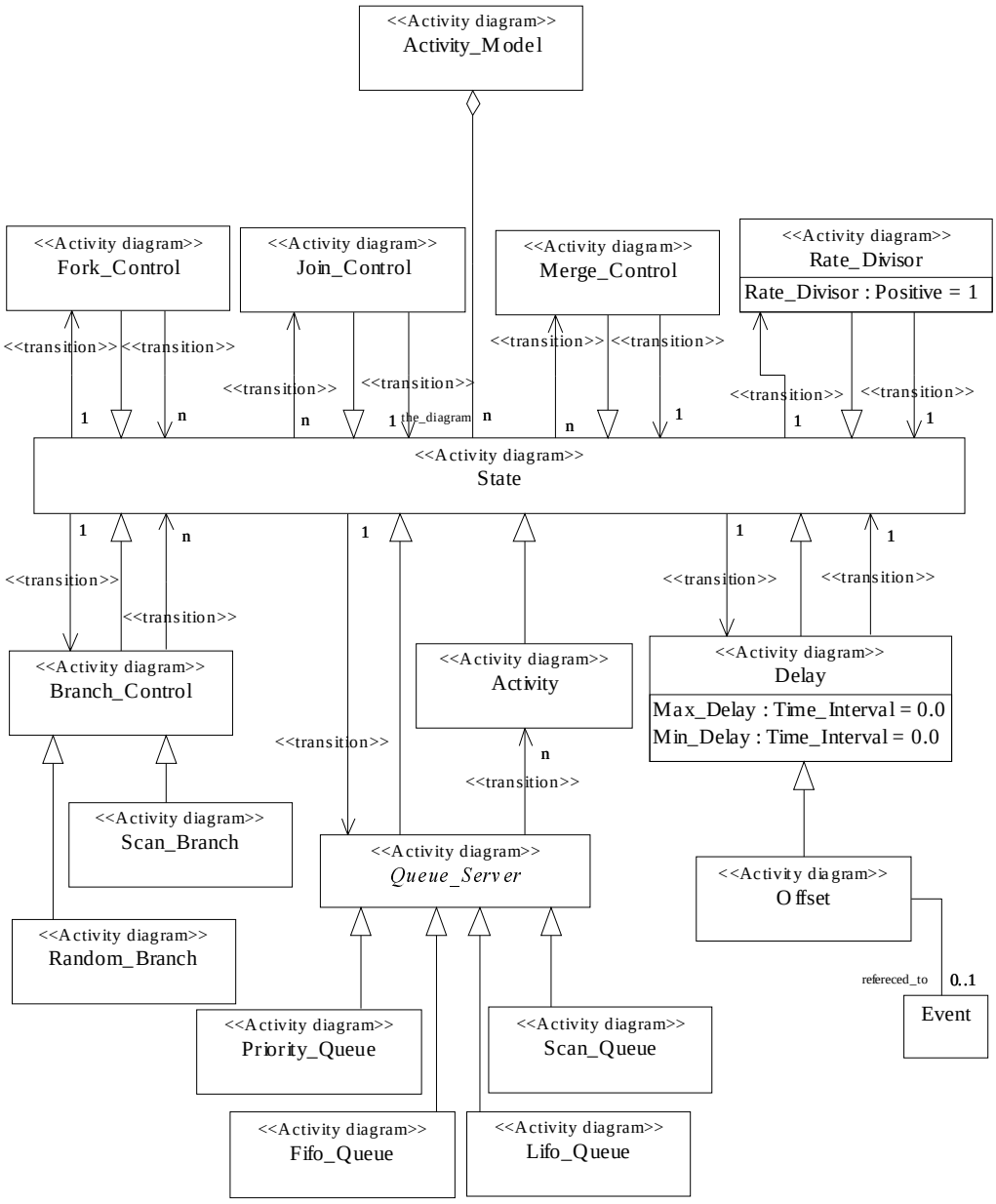


Diagrama 14.- Control activities classes.

Activity_Model

Describe el modelo de actividad (Activity_Model) de una Transaction (o Job).

El modelo de actividad esta compuesto de un conjunto de diagramas de actividad que describen las secuencias de actividades, estados y transiciones que se desencadenan como consecuencia de un patrón de eventos externos de entrada.

State

Son las activity states y action states UML del diagrama de actividad que representan los estados que alcanza el job en su ejecución.

Branch_Control

Representa la transición desde un estado de partida a uno de varios estados optativos de salida. Se representa mediante un rombo con una única transición de entrada y múltiples transiciones de salida.

El tipo de Branch_Control no es relevante a efecto de análisis de tiempo real (análisis de peor caso), solo se tiene en consideración en las herramientas de animación.

Derived from: State

Scan_Branch

Branch_Control en el que las transiciones de salida se generan siguiendo el orden con que están declaradas.

Derived from: Branch_Control

Random_Branch

Branch_Control en el que la transición de salida que se ha de realizar se elige aleatoriamente (todas con igual probabilidad) de entre las posibles.

Derived from: Branch_Control

Fork_Control

Representa la activación de varias líneas de flujo de control concurrentes a partir de una línea de flujo de control de entrada.

Se representa mediante una barra de sincronización con una única transición de entrada y múltiples transiciones de salida. Cuando se produce la transición de entrada se generan concurrentemente todas las transiciones de salida

Derived from: State

Join_Control

Representa la convergencia y unificación de varias líneas de flujo de control que constituyen entradas alternativas hacia una nueva secuencia de actividades comunes.

Dada la naturaleza concurrente y conducida por eventos de los sistemas que se modelan, no necesariamente las líneas de flujo de control proceden de un previo branching, sino que pueden proceder de líneas de flujo de control procedentes de eventos externos diferentes.

En el diagrama de actividad, un componente Join_Control se representa por un rombo con múltiples transiciones de entrada y una única transición de salida.

Derived from: State

Merge_Control

Representa un mecanismo de sincronización de múltiples líneas de flujo de control concurrentes hacia una única línea de flujo de control de salida (Rendezvous múltiple). Cuando todas las transiciones de entrada se han producido se produce una transición de salida. En un diagrama de actividad se representa por una barra de sincronismo con múltiples transiciones de entrada y una única transición de salida.

Un Merge_Control es un componente dinámico que recuerda las activaciones ocurridas en cada transición de entrada. Cuando el Merge_Control se activa, consume solo una de las activaciones de cada una de sus entradas, y sigue recordando las que restan, que continúan pendientes. Por ello, en cualquier modelo se requiere que sean idénticas las frecuencia de activación de cada una de las transiciones de entrada en un Merge_Control.

Derived from: State

Queue_Server

Es una activity con una única transición de entrada y múltiples transiciones de salida. Las transiciones de salida de una Queue_Server deben ser Activities que ejecutan una Operation o que ejecutan un Job que inicialmente ejecuta una Activity que ejecuta una Operation.

Un Queue_Server es de naturaleza dinámica, recuerda las activaciones de entrada, y por cada una de ellas produce una activación de salida en sólo una de las transiciones de salida que tiene. Una transición de salida sólo ocurre si la actividad cliente está dispuesta para la activación (esto es, ha terminado su activación anterior). Si varias actividades clientes de una Queue_Server están dispuestas para activación, cuando esta tiene pendiente una transición de salida, sólo se activa una de ellas, y esta se selecciona de acuerdo con una política de atención (Priority, Fifo, Lifo o Scan) establecida.

El tipo de Queue_Server que se declara no tiene influencia sobre los análisis de tiempo real (peor caso), sólo afecta a las herramientas de animación.

Derived from: State

Scan_Queue

Es una Queue_Server en la que se activa la actividad cliente que sigue a la última que fue servida, siguiendo el orden de declaración de las transiciones de salida.

Derived from: Queue_Server

Priority_Queue

Es una Queue_Server en la que se atiende la actividad cliente de mayor prioridad.

Derived from: Queue_Server

Fifo_Queue

Es una Queue_Server en la que se activa la actividad cliente que lleva mas tiempo dispuesta para activación.

Derived from: Queue_Server

Lifo_Queue

Es una Queue_Serve en la que se activa la actividad cliente que mas recientemente ha pasado a dispuesta para activación.

Derived from: Queue_Server

Activity

Es un Activity_State que representa una nueva ejecución de una Operation que se declara en su sentencia do/. El estado se abandona cuando se termina la ejecución de la Operation, realizándose entonces la transición de salida.

Sólo puede contener una sentencia do/.

Dado que en general los sistemas que se modelan son concurrentes, puede activarse una Activity antes que haya concluido su ejecución anterior. Cuando esto ocurre, habrá dos instancias de la Activity dispuestas para ser planificadas en el mismo Scheduling_Server. Ambas instancias comparten una misma política de planificación (y en su caso, tienen la misma prioridad).

La Operation que se referencia mediante su identificador en la sentencia do/ debe haber sido declarada y descrita en otra sección del modelo.

Las Activity de un Job_Model tienen una única transición de entrada y una única transición de salida.

Derived from: State

Rate_Divisor

Es una actividad dinámica que realiza una transición de salida por cada cierto número (Rate_Divisor) de transiciones de entrada que hayan ocurrido. En el diagrama se representa mediante una activity state con estereotipo Rate_Divisor. El valor del atributo Rate_Divisor se establece mediante una sentencia:

do/ Rate_Divisor(Rd : Positive = 1)

Derived from: State

Private Attributes:
Rate_Divisor : Positive = 1

Delay

Es una actividad que realiza una transición de salida un tiempo especificado después de que se haya producido su transición de entrada. Se representa como una activity state con estereotipo Delay y se establecen los tiempos que caracterizan el retraso mediante sentencias:

do/Max_Delay(D : Time_Interval = 0.0)
do/Min_Delay(D : Time_Interval = 0.0)

Derived from: State

Private Attributes:
Max_Delay : Time_Interval = 0.0
Min_Delay : Time_Interval = 0.0

Offset

Es una actividad que realiza una transición de salida al menos un tiempo especificado después de que se haya producido el evento al que se hace referencia. Para que la transición de salida se produzca es necesario que se haya realizado previamente la transición de entrada. Se representa por una activity state con estereotipo Offset. Los tiempos que caracterizan el retraso se establece a través de las sentencias:

do/ Max_Delay(D : Time_Interval = 0.0)
do/ Min_Delay(D : Time_Interval = 0.0)

y el evento al que hace referencia se establece a través de una sentencia:

do/ Reference_To(Ev : Identifier)

Si el evento referenciado es externo, Event_Identifier debe ser el identificador de la External_Event_Source que caracteriza su generación y si el evento referenciado es relativo a alcanzar un estado interno, el identificador debe ser el del correspondiente Named_State.

Derived from: Delay

Event

Es una clase abstarcta que generaliza de forma compatible los eventos externos (definidos por External_Event_Source) y los eventos internos (Named_State).

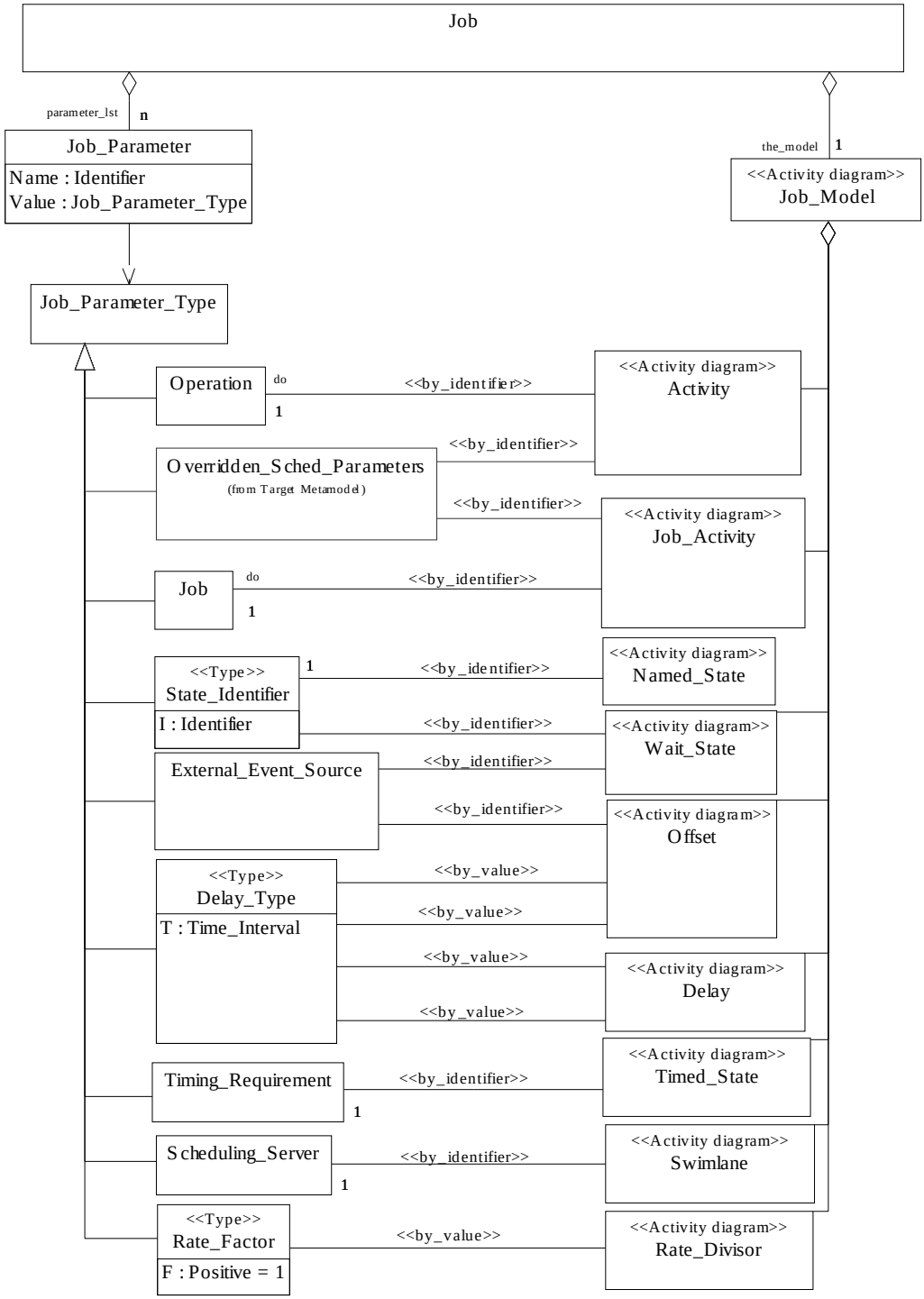


Diagrama 15.- Job Parameter classes.

Job

Modela la actividad potencialmente concurrente que se desencadena cuando se ejecuta un componente lógico (procedimiento, función método, etc.) en un programa concurrente. La actividad modelada por un job puede requerir que intervengan múltiples threads y puede perdurar aun después de que haya concluido la invocación del componente que la desencadenó.

Derived from: Job_Parameter_Type

Job_Parameter

Los parámetros que se definen en un Job se declaran como atributos de la clase que define el Job. Los parámetros se declaran con su identificador, su tipo y en su caso, el valor por defecto.

Private Attributes:

Name : Identifier

Nombre del parámetro del Job. En el modelo se representa por el identificador del atributo con que se declara el parámetro.

Value : Job_Parameter_Type

Valor que se asigna al parámetro en una invocación del job. El valor se asigna con una sentencia del tipo genérico:

do/ Job_name(Param_Ident=> Param_Value)

Job_Parameter_Type

Los parámetros de un job deben tener un tipo declarado explícitamente. Los tipos admitidos son:

- Operation
- Job
- State_Identifier
- External_Event_Source
- Delay_Type
- Timing_Requirement
- Scheduling_Server
- Rate_Factor
- Overridden_Sched_Parameter

Operation

Modela la temporización de la ejecución de un segmento de código que se ejecuta dentro de un único scheduling server (thread). La característica específica del componente lógico modelado por una Operation es que en su código no existe ningún componente de sincronización con otro thread.

Derived from: **Primitive_Operation,**
Oper_Parameter_Type,
Job_Parameter_Type

State_Identifier

Tipo de parámetro de un job que representa el identificador de un componente (Named_State o Wait_State) utilizado en su descripción.

Derived from: Job_Parameter_Type

Private Attributes:

I : Identifier

External_Event_Source

Representa una secuencia (stream) de eventos que tienen su origen externo al código del sistema. Pueden ser originados por el entorno o por componentes hardware del sistema (timer, etc). Lo específico de su patrón de generación es que es autónomo y no depende del estado o de la evolución del sistema que se está modelando.

Derived from: Event,
Job_Parameter_Type

Delay_Type

Tipo de parámetro de un job que generaliza los argumentos del tipo Time_Interval que pueden proporcionarse como argumento en sentencias do/ propias de los Offset y Delay.

Derived from: Job_Parameter_Type

Private Attributes:

T : Time_Interval

Timing_Requirement

Representa el requerimiento temporal que se asocia a un estado (State_Action) dentro del contexto de una transacción.

Cada requerimiento temporal se define como relativo a un evento externo o a una transición que se encuentra dentro de la línea directa de flujo que ha provocado la transición a la que se asocia el requerimiento temporal.

Cuando una transición puede resultar como unión (join) de diferentes líneas de flujo de control con origen en diferentes fuentes de eventos externos, la restricción temporal asociada a ella sólo se aplica a los eventos originados por la fuente de eventos externos a la que hace referencia el requerimiento temporal.

Los diferentes requerimientos temporales de un requerimiento compuesto se aplican con criterio conjuntivo (and) si son relativos a un mismo evento y con carácter disyuntivo (or) si es relativo a diferentes eventos.

Derived from: **Job_Parameter_Type**

Scheduling_Server (From Target Metamodel)

Modela un proceso activo simple (con un solo thread) dentro del que se ejecutan una o varias actividades secuencialmente y que es ejecutado por el recurso de procesamiento que tiene referenciado.

Así mismo, modela la política de planificación y la capacidad de procesamiento disponible para ejecutar las actividades que se le asignan.

Derived from: **Job_Parameter_Type**

Rate_Factor

Tipo de parámetro de un job que establece el valor a asignar al atributo de un Rate_Divisor de su descripción.

Derived from: Job_Parameter_Type

Private Attributes:
F : Positive

Job_Model

Diagrama de actividad que describe las actividades, acciones y scheduling_server, de que se compone un job.

El Job_Model es un Activity_Model ampliado con Conector_State.

Derived from: **Activity_Model**

Activity

Es un Activity_State en el que ser alcanzado representa que se activa una nueva ejecución de una Operation que se declara en su sentencia do/. El estado se abandona cuando se termina la ejecución de la Operation, realizándose entonces la transición de salida.

Sólo puede contener una sentencia do/.

Dado que en general los sistemas que se modelan son concurrentes, puede activarse una Activity antes que haya concluido su ejecución anterior. Cuando esto ocurre, habrá dos instancias de la Activity dispuestas para ser planificadas en el mismo

Scheduling_Server. Ambas instancias comparten una misma política de planificación (y en su caso, tienen la misma prioridad).

La Operation que se referencia mediante su identificador en la sentencia do/ debe haber sido declarada y descrita en otra sección del modelo.

Las Activity de un Job_Model tienen una única transición de entrada y una única transición de salida.

Derived from: State

Job_Activity

Es un Activity_State que representa la activación de una nueva ejecución del job que se declara en su sentencia do/.

Derived from: State

Wait_State

Es un action state en el que al ser alcanzado se suspende la correspondiente línea de flujo de control, a la espera de que se genere un evento externo o de que se alcance un Named_State en alguna sección del modelo de la transacción.

Si el Wait_State representa una espera a un evento externo, el Identificador del Wait_State debe ser el mismo que el identificador del External_Event_Source que caracteriza la temporización de los eventos externos y que debe haber sido declarado en la transacción.

Si el Wait_State representa una espera a que el modelo alcance un determinado estado, el identificador del Wait_State debe coincidir con el del Named_State a cuya espera se suspende. La combinación Named_State y Wait_State con el mismo identificador es el mecanismo que se dispone para transferir una línea de flujo de control concurrente de una sección de la transacción a otra.

Se impone la restricción de que solo puede existir un Wait_State por cada Named_State definido en el modelo del job.

Derived from: State

Offset

Es una actividad que realiza una transición de salida al menos un tiempo especificado después de que se haya producido el evento al que se hace referencia. Para que la transición de salida se produzca es necesario que se haya realizado previamente la

transición de entrada. Se representa por un activity state con estereotipo Offset. Los tiempos que caracterizan el retraso se establecen a través de las sentencias:

do/ Max_Delay(D : Time_Interval = 0.0)
do/ Min_Delay(D : Time_Interval = 0.0)

y el evento al que hace referencia se establece a través de una sentencia:

do/ Reference_To(Ev : Identifier)

Si el evento referenciado es externo, Event_Identifier debe ser el identificador de la External_Event_Source que caracteriza su generación y si el evento referenciado es relativo a alcanzar un estado interno, el identificador debe ser el del correspondiente Named_State.

Derived from: Delay

Delay

Es una actividad que realiza una transición de salida un tiempo especificado después de que se haya producido su transición de entrada. Se representa como una activity state con estereotipo Delay y se establecen los tiempos que caracterizan el retraso mediante sentencias:

do/Max_Delay(D : Time_Interval = 0.0)
do/Min_Delay(D : Time_Interval = 0.0)

Derived from: State

Private Attributes:
Max_Delay : Time_Interval = 0.0
Min_Delay : Time_Interval = 0.0

Timed_State

Representa una action state que se declara para asignar un requerimiento temporal (Timing requirement) al estado que representa dentro del job.

El nombre del Timed_State debe tener un identificador idéntico al del Timing_Requirement que se le asocia, el cual debe estar declarado en la transacción a la que corresponden ambos.

Un Timed_Sate tiene siempre una transición de entrada y puede tener una o ninguna transición de salida.

Derived from: State

Swimlane

Banda vertical del diagrama de actividad que representa a cada Scheduling_Server que participa en la ejecución del job.
Cada Activity del Job debe situarse en la Swimlane que corresponde al Scheduling_Server al que se asigna su ejecución El Scheduling_Server en que se ejecutan las actividades declaradas en un Job se especifica en la propia descripción del Job.
Las Activity que en la descripción de un Job se asignan a un Swinlane sin nombre, se ejecutan en el Scheduling_Server en que se encuentra la Activity que invoca al Job.

Named_State

Representa un action state (y por tanto de duración nula) del Job_Model, al que se le asigna un nombre para poder referenciarlo dentro de la misma transacción. Un Named_State se utiliza en el modelo de actividad en los siguientes dos casos:

- Cuando se necesita hacer referencia al estado que representa fuera del modelo del Job. Esto ocurre, cuando se necesita que una de las líneas concurrentes de flujo de control del Job continúe en otra sección del modelo de la misma transacción.
- Para finalizar una línea de flujo concurrente que haya permanecido activa después de que haya finalizado la línea que corresponde a la invocación del Job.

El ámbito del identificador de un Named_State es la instancia del Job en que está declarado. Los Named_States que están declarado en un Job, se instancian como Named_State diferentes en cada instanciación del Job. Si diferentes Job necesitan hacer referencia a un mismo Name_State, este debe ser transferido a través de un parámetro. Un Named State se declara tanto en un Named_State del diagrama de actividad, como en un Wait_State que hace referencia al Named_State.

Un Name_State tiene siempre una transición de entrada y puede tener o no una transición de salida.

Derived from: **State,**
Event

Rate_Divisor

Es una actividad dinámica que realiza una transición de salida por cada cierto número (Rate_Divisor) de transiciones de entrada que hayan ocurrido. En el diagrama se representa mediante una activity state con estereotipo Rate_Divisor. El valor del atributo Rate_Divisor se establece mediante una sentencia:
do/ Rate_Divisor(Rd : Positive = 1)
Derived from: State

Private Attributes:
Rate_Divisor : Positive = 1

Scenarios Metamodel

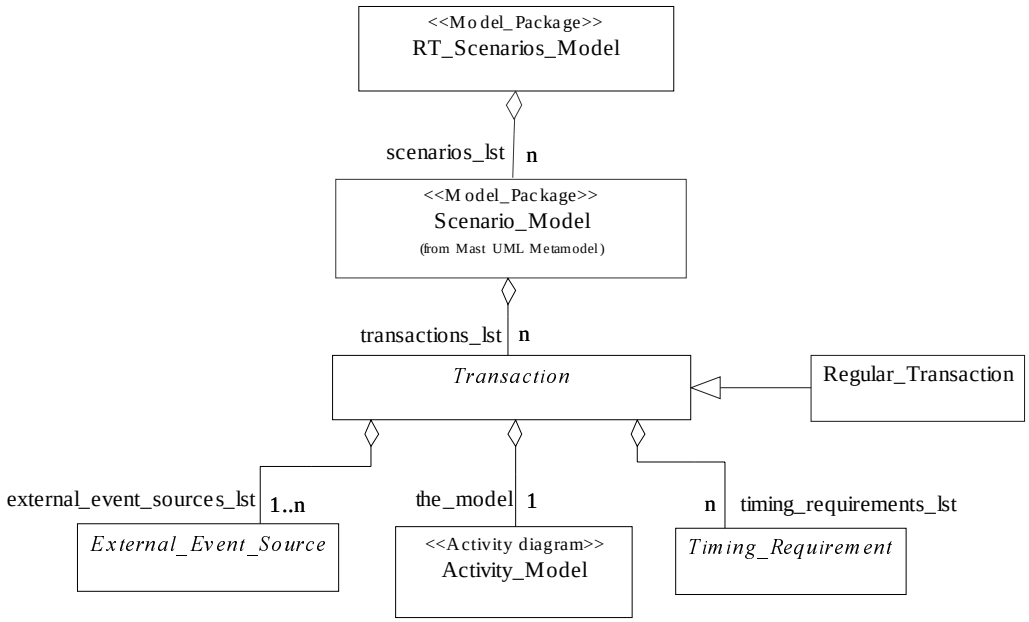


Diagrama 16: RT Scenarios Model.

RT_Scenarios_Model (From UML Mast Metamodel)

Conjunto de configuraciones o de modos de operación del sistema en los que hay definidas especificaciones de tiempo real.

Se implementa como un paquete que debe estar definido directamente dentro del paquete Mast_RT_View.

Scenario_Model

Modela cada configuración o modo de operación que puede ser alcanzado por el sistema en los que están establecidos requerimientos de tiempo real.

Representa una carga concreta de trabajo (workload) en la que deben satisfacerse un conjunto de requerimientos de tiempo real. Está constituido por el conjunto de los modelos de las transacciones que concurren en él.

En un modelo cada Scenario_Model se representa mediante un package con el nombre del escenario y se define en el package RT_Scenarios_Model.

Transaction

Describe la secuencia no iterativa de actividades que se desencadenan como respuesta a un patrón de eventos externos que sirve de marco para definir los requerimientos temporales. Cada transacción incluye todas las actividades que se desencadenan como consecuencia del patrón de entrada y todas la actividades que requieren sincronización directa con ellas (intercambio de eventos, sincronización por invocación, etc.).

El patrón de disparo de una transacción está constituido por el conjunto de los eventos externos que son la causa de que la transacción se lleve a cabo de forma completa.

El conjunto de las transacciones que constituyen el modelo del sistema proporciona una descripción complementaria a la descripción funcional del sistema convencional. Con el modelo Mast se explicitan la temporalidad, la dependencia de flujo y la concurrencia entre las actividades de la aplicación.

Una transacción incluye todas las actividades interdependientes por transferencias de flujo o sincronización. Por lo que los conjuntos de componentes de las diferentes transacciones son disjuntos entre sí y no intercambian eventos entre ellos.

Entre las diferentes transacciones de un sistema existen las relaciones y dependencias que se establecen debido a que las diferentes actividades de unas y de otras compiten por acceder al uso en régimen de exclusividad de los recursos comunes (Processing Resource y Shared_Resource) que puedan existir.

Aunque las transacciones son secuencias abiertas (no iterativas) de actividades, estas pueden ser activadas periódicamente y pueden solaparse en el tiempo sucesivas ejecuciones de ellas. Esto es lo habitual en sistemas que operan en modo pipeline.

Regular_Transaction

Representa una transacción analizable mediante las herramientas Mast. Las restricciones que debe satisfacer la descripción de la Regular_Transaction son:

a) Cada transacción debe tener asociado, al menos, un evento externo.

b) En la transacción deben estar incluidas todas las actividades que sean activadas en las líneas de flujo de control que tienen su origen en las fuentes de eventos externos de la transacción.

c) En la transacción deben estar incluidas todas las actividades que estén directamente sincronizadas (por invocación, por transferencia de eventos, o por componentes activos de sincronización) con actividades de la transacción. Esta característica hace que los conjuntos de actividades de las diferentes transacciones de un escenario sean disjuntos.

d) No pueden existir dependencias circulares de activación, esto es, no pueden existir bucles en los diagramas de actividad que describen una transacción.

e) Todos los recursos compartidos reservados en la transacción deben ser liberados en ella.

f) No pueden converger en dos ramas de un Merge_Control líneas de flujo de control activadas por diferentes External_Event_Source.

g) No pueden utilizarse componentes de la clase Rate_Divisor en líneas de flujo de control resultantes de la unión (join) de líneas de flujo de control procedentes de diferentes External_Event_Source.

Derived from: **Transaction**

Activity_Model

Describe el modelo de actividad (Activity_Model) de una Transaction o Job.

El modelo de actividad esta compuesto de un conjunto de diagramas de actividad que describen las secuencias de actividades, estados y transiciones que se desencadenan como consecuencia de un patrón de eventos externos de entrada.

External_Event_Source

Representa una secuencia (stream) de eventos que tienen su origen externo al código del sistema. Pueden ser originados por el entorno o por componentes hardware del sistema (timer, etc). Lo específico de su patrón de generación es que es autónomo y no depende del estado o de la evolución del sistema que se está modelando.

Derived from: **Event,**
Job_Parameter_Type

Timing_Requirement

Representa el requerimiento temporal que se asocia a un estado (State_Action) dentro del contexto de una transacción.
Cada requerimiento temporal se define como relativo a un evento externo o a una transición que se encuentra dentro de la línea directa de flujo que ha provocado la transición a la que se asocia el requerimiento temporal.
Cuando una transición puede resultar como unión (join) de diferentes líneas de flujo de control con origen en diferentes fuentes de eventos externos, la restricción temporal asociada a ella sólo se aplica a los eventos originados por la fuente de eventos externos a la que hace referencia el requerimiento temporal.

Los diferentes requerimientos temporales de un requerimiento compuesto se aplican con criterio conjuntivo (and) si son relativos a un mismo evento, y con carácter disyuntivo (or) si es relativo a diferentes eventos.

Derived from: Job_Parameter_Type

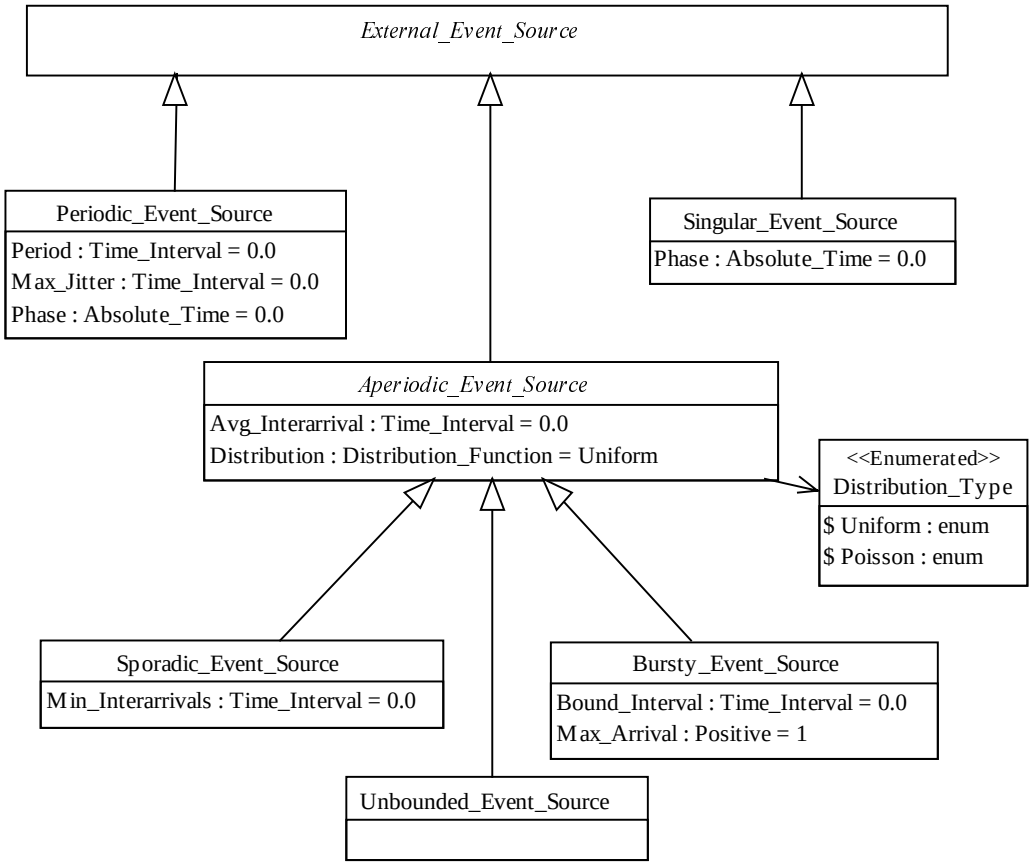


Diagrama 17:External Event Source classes.

External_Event_Source

Representa una secuencia (stream) de eventos que tienen su origen externo al código del sistema. Pueden ser originados por el entorno o por componentes hardware del sistema (timer, etc). Lo específico de su patrón de generación es que es autónomo y no depende del estado o de la evolución del sistema que se está modelando.

Derived from: **Event,**
Job_Parameter_Type

Periodic_Event_Source

Representa una secuencia con un patrón de generación aproximadamente periódico. Está caracterizado por su periodo (Period), su máximo jitter (Max_Jitter) y el tiempo de generación del primer evento (Phase).

Derived from: External_Event_Source

Private Attributes:
Period : Time_Interval = 0.0
Periodo de generación de eventos.
Max_Jitter : Time_Interval = 0.0
Máximo jitter de los eventos respecto del ritmo periódico.
Phase : Absolute_Time = 0.0
Fase de generación del primer evento.

Aperiodic_Event_Source

Modela una fuente de eventos que los genera de forma aperiódica, esto es, con un patrón aleatorio de tiempos entre eventos.

Derived from: External_Event_Source

Private Attributes:
Avg_Interarrival : Time_Interval = 0.0
Valor promedio de los tiempos entre eventos consecutivos.
Distribution : Distribution_Function = Uniform
Tipo de función de distribución de los tiempos entre eventos consecutivos.

Distribution_Function

Tipos de función de distribución de eventos no periódicos.

Private Attributes:
Uniform : enum
Distribución uniforme. Todos los tiempos dentro del rango establecido tienen igual probabilidad.
Poisson : enum
Los tiempos entre eventos tienen una distribución de tipo Poisson.

Sporadic_Event_Source

Los intervalos de tiempo entre eventos consecutivos están acotados inferiormente por un tiempo mínimo establecido (Min:Interarrivals).
Derived from: Aperiodic_Event_Source

Private Attributes:
Min_Interarrivals : Time_Interval = 0.0
Tiempo mínimo que se puede presentar entre dos eventos consecutivos.

Unbounded_Event_Source

Generador de eventos esporádicos que poseen un patrón de tiempos entre eventos aleatorio pero en el que los tiempos entre eventos consecutivos no están acotados inferiormente.

Derived from: Aperiodic_Event_Source

Bursty_Event_Source

Son eventos esporádicos que se producen en grupos. El número de eventos en un grupo esta superiormente acotado a un número establecido (Max_Arrivals), y los tiempos de los intervalos entre eventos dentro del grupo esta limitado por una cota inferior conocida (Bound_Interval).

Derived from: Aperiodic_Event_Source

Private Attributes:
Bound_Interval : Time_Interval = 0.0
Tiempo mínimo que se puede presentar entre dos ráfagas de eventos consecutivos dentro de un grupo de eventos.
Max_Arrival : Positive = 1
Máximo número de eventos en un grupo.

Singular_Event_Source

Evento que sólo se genera una única vez (evento singular). Se utiliza para describir el instante en que se produce una situación singular como arranque, cambio de modo, etc.
Derived from: External_Event_Source

Private Attributes:
Phase : Absolute_Time = 0.0
The phase represents the time at which the event is generated.

El origen de tiempo respecto del que se formula la fase no está definido, pero la fase de todos los eventos singulares está referida el mismo origen de tiempo. Sólo la diferencia de fase entre eventos singulares es relevante.

Si se necesita tener un origen absoluto de tiempo definido, se define un evento singular con Phase=0.0

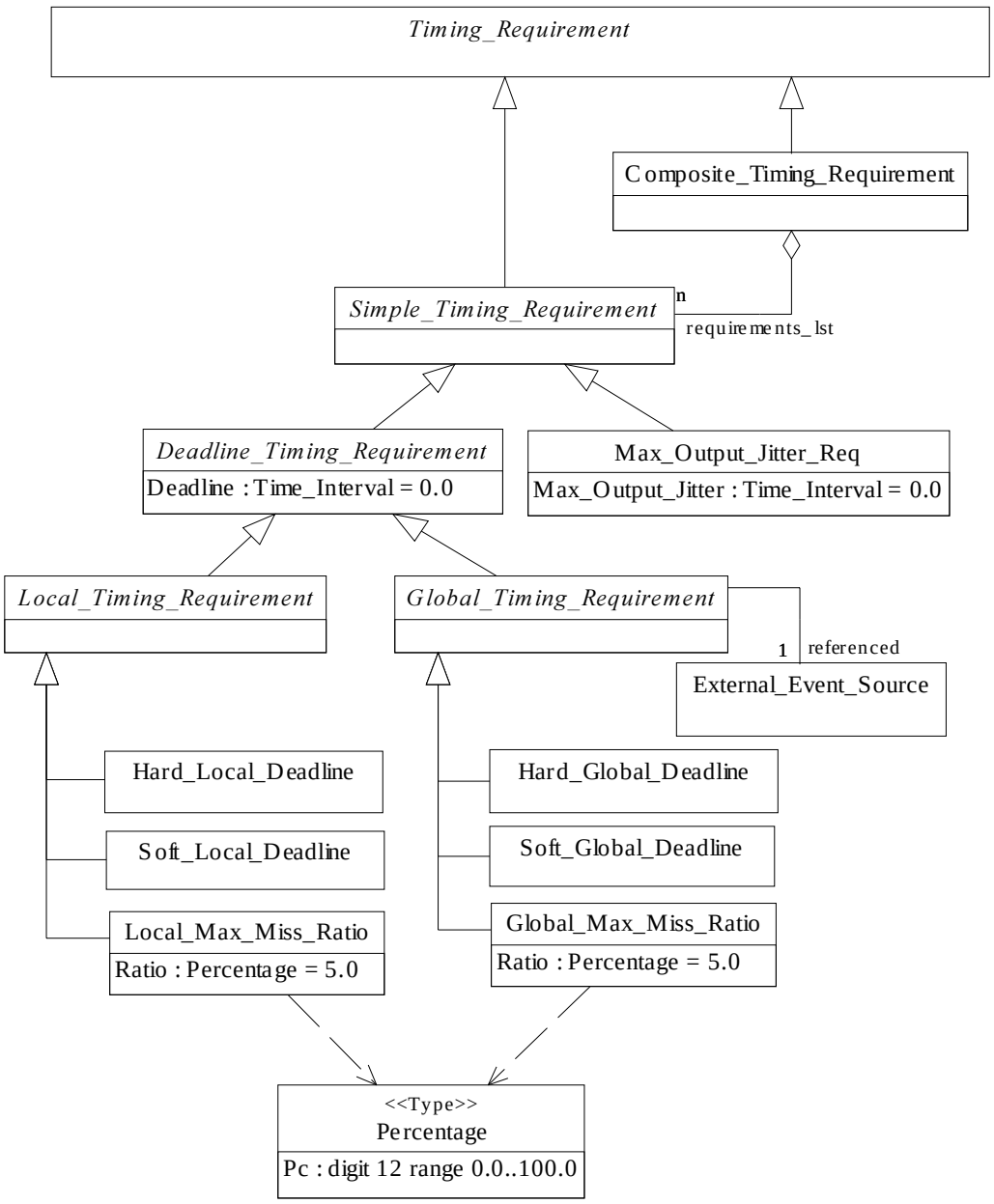


Diagrama 18: Timing Requirements classes.

Timing_Requirement

Representa el requerimiento temporal que se asocia a un estado (State_Action) dentro del contexto de una transacción.

Cada requerimiento temporal se define como relativo a un evento externo o a una transición que se encuentra dentro de la línea directa de flujo que ha provocado la transición a la que se asocia el requerimiento temporal.
Cuando una transición puede resultar como unión (join) de diferentes líneas de flujo de control con origen en diferentes fuentes de eventos externos, la restricción temporal asociada a ella sólo se aplica a los eventos originados por la fuente de eventos externos a la que hace referencia el requerimiento temporal.
Los diferentes requerimientos temporales de un requerimiento compuesto se aplican con criterio conjuntivo (and) si son relativos a un mismo evento y con carácter disyuntivo (or) si es relativo a diferentes eventos.

Derived from: **Job_Parameter_Type**

Simple_Timing_Requirement

Requerimiento temporal simple con una única restricción temporal.

Derived from: **Timing_Requirement**

Composite_Timing_Requirement

Conjunto de requerimientos simples que se aplican a un mismo estado.

Cada requerimiento temporal es siempre relativo a un evento externo o a una transición que se encuentra dentro de la línea directa de flujo que ha provocado la transición a la que se asocia el requerimiento temporal.

Cuando una transición puede resultar como unión (join) de diferentes líneas de flujo de control con origen en diferentes fuentes de eventos externos, la restricción temporal asociada a ella solo se aplica a los eventos originados por la fuente de eventos externos a la que hace referencia el requerimiento temporal.

Derived from: **Timing_Requirement**

Deadline_Timing_Requirement

Requerimiento temporal que se refiere al plazo en el que se debe alcanzar el estado al que está asociado.

Derived from: **Simple_Timing_Requirement**

Private Attributes:
Deadline : Time_Interval = 0.0
Plazo referido.

Local_Timing_Requirement

Requerimiento temporal local. Establece requerimientos temporales sobre la duración de la actividad cuya finalización genera el evento.

Derived from: Deadline_Timing_Requirement

Hard_Local_Deadline

Plazo local estricto. Requiere que la duración de la actividad cuya finalización genera el evento asociado sea inferior al plazo establecido (Deadline).

Derived from: Local_Timing_Requirement

Soft_Local_Deadline

Plazo local laxo. Requiere que el promedio de los tiempos de duración de la actividad cuya finalización genera el evento sea inferior al plazo establecido (Deadline).

Derived from: Local_Timing_Requirement

Local_Max_Miss_Ratio

Plazo local laxo con tanto por ciento de fallos limitado. Requiere que para un tanto por ciento de casos establecido (Ratio) se satisfaga que la duración de la actividad cuya finalización genera el evento al que está asociada la restricción temporal, sea inferior al plazo establecido (Deadline).

Derived from: Local_Timing_Requirement

Private Attributes:
Ratio : Percentage = 5.0
Tanto por ciento máximo de fallos permitido.

Global_Timing_Requirement

Requerimiento temporal global que establece condiciones de temporización para la ocurrencia del evento interno al que se asocia con referencia a la ocurrencia del evento al que hace referencia (Referenced).

Derived from: Deadline_Timing_Requirement

External_Event_Source

Representa una secuencia (stream) de eventos que tienen su origen externo al código del sistema. Pueden ser originados por el entorno o por componentes hardware del sistema (timer, etc). Lo específico de su patrón de generación es que es autónomo y no depende del estado o de la evolución del sistema que se está modelando.

Derived from: **Event,**
Job_Parameter_Type

Hard_Global_Deadline

Plazo global estricto. Requiere que el evento al que está asociado se produzca antes de que transcurra el plazo establecido (Deadline) desde que se produjo el evento de referencia (Referenced).

Derived from: Global_Timing_Requirement

Soft_Global_Deadline

Plazo global laxo. Requiere que el promedio del tiempo que transcurre entre los eventos a los que está asociado el requerimiento temporal y los correspondientes eventos del tipo referenciado, sea inferior al plazo establecido (Deadline).

Derived from: Global_Timing_Requirement

Global_Max_Miss_Ratio

Plazo global laxo con tanto por ciento de fallos limitado. Requiere que para un tanto por ciento de casos establecido (Ratio) se satisfaga que el retraso entre el evento al que está asociada la restricción temporal y el evento de referencia (referenced) sea inferior al plazo establecido (Deadline).

Derived from: Global_Timing_Requirement

Private Attributes:
Ratio : Percentage = 5.0
Tanto por ciento máximo de fallos permitido.

Percentage

Subtipo real que representa un tanto por ciento en el rango 0.0 .. 100.0.

Private Attributes:
Pc : digit 12 range 0.0..100.0

Max_Output_Jitter_Req

Requerimiento temporal que se refiere al jitter de los tiempos en que se alcanza ese estado.

Derived from: **Simple_Timing_Requirement**

Private Attributes:
Max_Output_Jitter : Time_Interval = 0.0
Máximo jitter admisible.

DEFINICION DE ESTEREOTIPOS UTILIZADOS EN EL METAMODELO.

Class Stereotypes

[No Stereotype]
Las metaclases sin estereotipo del metamodelo se instancian en los modelos como clases. El estereotipo de la clase instanciada indica la metaclase de la que procede. La información cuantitativa del componente que se modela se formula a través de los valores iniciales y los diagramas de actividad de la clase instanciada.

<<Model_Package>>
Las metaclases con este estereotipo representan a clases contenedoras utilizadas para la organización del modelo al nivel mas alto. Estas metaclases se instancian en los modelos como paquetes.

<<Activity_Diagram>>
Las metaclases con este estereotipo representan componentes Mast que se utilizan dentro de los diagramas de actividad que describen las operaciones, job o transacciones. En un modelo, sus instancias son activity states o action states de diagramas de actividad asociados a componentes del mismo.

Attribute Stereotypes

<<Enumerated>>
Representa un tipo de atributo formulado por enumeración de los valores que puede tomar.

<<Subtype>>
Representa un tipo de atributo descrito a partir de un tipo ya definido, que es compatible con él pero que tiene un rango de valores restringido.

<<Type>>
Representa un tipo de atributo definido a partir de un tipo básico (Integer, Natural, Float, String, etc.).

Association Stereotypes

<<by_identifier>>
Representa una asociación definida en el metamodelo que en el modelo se instancia mediante una referencia explícita al identificador de la clase con la que se establece la asociación.

<<by_value>>
Asociación del metamodelo entre un parámetro y un atributo de una clase explícitamente establecida por él, que se instancia en el modelo expresando directamente el valor numérico del atributo.

<<ordered>>
Asociación múltiple en la que el orden de los componentes asociados es relevante.

<<transition>>
Asociación del metamodelo entre metaclases con el estereotipo <<Activity diagram>> que en el modelo se instancian como transiciones del diagrama de actividad.

LISTA DE COMPONENTES