Actas de las

XIII Jornadas de Tiempo Real

(JTR 2010)

Granada, 4 y 5 de Febrero de 2010

EDITORES

Manuel I. Capel Tuñón
Juan Antonio Holgado Terriza

ORGANIZADO POR:

Grupo de Sistemas Concurrentes
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada

Reservados todos los derechos. Queda rigurosamente prohibida, sin la autorización escrita de los titulares del copyright, bajo las sanciones establecidas en las leyes, la reproducción total o parcial de esta obra por cualquier medio o procedimiento.

© Los autores de cada artículo incluido en estas actas.

ISBN 10: 978-84-92757-51-0

Depósito Legal: GR-414/2010

Maquetación: Juan Antonio Holgado Terriza

Fotocomposición: Ediciones Copicentro Granada SL

Impreso por Talleres Gráficos Copicentro Granada SL

Impreso en España - Printed in Spain

MAST 2: Nueva revisión del modelo de tiempo real.

J.M. Drake, M. González Harbour, J.J. Gutiérrez,
P. López Martínez, J. Medina y J.C. Palencia

Departamento de Electrónica y Computadores, Universidad de Cantabria,
39005-Santander, SPAIN
{drakej,mgh,gutierjj,lopezpa,medinajl,palencijc}@unican.es
{http://mast.unican.es/}

Resumen: Se describen y justifican los aspectos claves que se incluyen en la nueva revisión del modelo del entorno de análisis y desarrollo de tiempo real MAST.

Palabras clave: Real-time, RT-model, Schedulability analysis

1 Antecedentes[1]

MAST (Modeling and Analysis Suite for Real-Time Applications) nace en el año 2000 [1] como respuesta a la necesidad, que se planteó en las I Jornadas de Tiempo Real celebradas en Las Navas del Marqués, de disponer de un entorno de modelado de sistemas de tiempo real que constituyera un crisol y un banco de prueba en el que integrar las metodologías y herramientas que desarrollaran los diferentes grupos de investigación. Asimismo, se buscaba establecer un expositor en el que los grupos y empresas pudieran encontrar herramientas de libre uso. Visto con la perspectiva de los 10 años transcurridos, se ha satisfecho el primer objetivo y son muchos los grupos

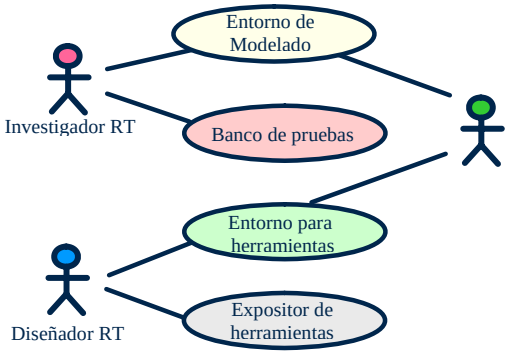

Fig. 1. Casos de uso de Mast

1. Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia del Gobierno de España dentro del proyecto TIN2008-06766-C03-03 (RT-MODEL) y por la Unión Europea bajo el proyecto FP7/NoE/214373 (ArtistDesign). Este trabajo refleja sólo el punto de vista de los autores; la UE no se responsabiliza del uso que se pueda hacer de la información contenida.

nacionales e internacionales que lo han utilizado como base para sus trabajos de investigación. Sin embargo, posiblemente por la incapacidad de mantener las herramientas siempre actualizadas, el segundo objetivo ha evolucionado de constituir un expositor de herramientas, a ser un punto de búsqueda donde encontrar a los expertos que pueden desarrollar las herramientas que se necesitan.

Como se muestra en la figura 2, el entorno MAST se compone de tres secciones:

- Un núcleo central constituido por el metamodelo que define los elementos de modelado y las formas en que estos elementos se pueden asociar entre sí para constituir el modelo de un sistema, así como los formatos en los que se codifican los modelos, los resultados de análisis y las trazas que describen la evolución del sistema modelado.
- Un conjunto de herramientas de análisis de planificabilidad y de herramientas de diseño que ayudan al diseñador a encontrar los parámetros de configuración para la planificación del sistema que se desarrolla. Algunas de estas herramientas están especializadas en algún aspecto específico, y otras están coordinadas para cubrir las sucesivas fases de un proceso de diseño.
- Un conjunto de herramientas para la construcción de los modelos de tiempo real que necesita el diseñador en los procesos de desarrollo de las aplicaciones. Algunas herramientas son simplemente interfaces gráficas para manejar los modelos y los resultados que se generan, y otras, generan los modelos MAST partiendo de modelos con otros niveles de abstracción.

En la página WEB *{http://mast.unican.es}* de soporte de MAST se pueden encontrar las especificaciones y las herramientas que actualmente constituyen el entorno.

El modelo de tiempo real que utiliza MAST es de naturaleza reactiva y se compone de dos descripciones superpuestas del sistema. Por un lado, la descripción del sistema como una secuencia de actividades que se ejecutan concurrentemente como respuesta

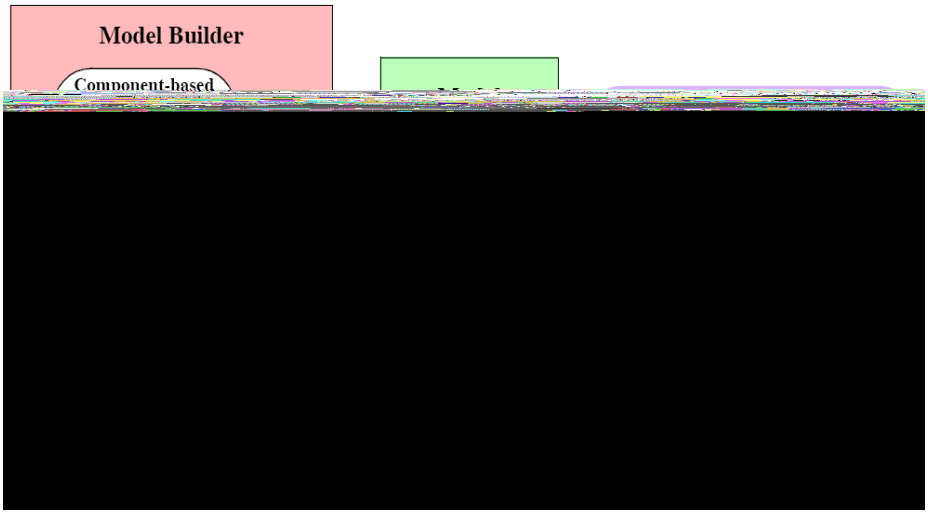

Fig. 2. Secciones del entorno MAST

a los eventos del entorno o temporizados que recibe el sistema, y por otro, la descripción de los procesos de sincronización que deben realizarse antes de las ejecuciones de esas actividades cuando se accede a recursos que requieren exclusión mutua, y que en su acceso introducen bloqueos o suspensiones que incrementan las latencias de las respuestas. Aunque existen otras metodologías de modelado de sistemas de tiempo real, la utilizada por MAST es la más implantada por tener dos importantes apoyos en su favor: la tecnología con la que se construyen los sistemas de tiempo real se basa en sistemas operativos con threads, planificadores y mecanismos de sincronización que tienen correspondencia directa en las técnicas de modelado de tiempo real que se utilizan; y además, las dos principales especificaciones estándares SPT [2] y MARTE [3] que han sido propuestas por OMG como base para el desarrollo de herramientas de tiempo real, también siguen esta metodología.

2 Motivación de la revisión Mast 2.

Tres han sido los motivos que han inducido a realizar en este momento la nueva revisión importante de MAST:

1. **Alinear MAST con el estándar MARTE**. Este es un punto muy importante para la difusión de MAST en muchos ámbitos, ya que permite rentabilizar el esfuerzo de modelar con MAST, al ser en gran parte compatible con los modelos de otras herramientas basadas en el estándar, y así mismo, el conocimiento del estándar ayuda a utilizar MAST. La alineación está facilitada por la cercanía conceptual de MAST con el contenido de las secciones *Schedulability Analysis Modeling(SAM)* y *Generic Resource Modeling (GRM)* de MARTE, lo cual es consecuencia de que los redactores de esas secciones conocían y usaban MAST.
2. **Incrementar la capacidad de modelado de MAST**. Los sistemas de tiempo real han evolucionado en estos años y nuevas infraestructuras, estrategias y recursos han aparecido y hacia ellas se han desplazado las líneas de investigación. Los aspectos que han sido considerados son:
   - Diseño de tiempo real basado en reservas de recursos.
   - Diseño de aplicaciones basado en componentes.
   - Estrategias basadas en tiempo compartido y de planificación basada en tablas de tiempos.
   - Redes de tiempo real complejas con conmutadores (*switches*) y enrutadores (*routers*).
   - Sistemas con múltiples relojes y sistemas distribuidos con procesadores sincronizados.

   Todas ellas han requerido introducir nuevos elementos raíces de modelado, y extender otros de los que ya existían en las versiones anteriores de MAST.
3. **Dar consistencia y uniformizar a MAST**, frente a las diferentes adendas que habían sido introducidas en las subversiones previas.

3 Alineación con la especificación MARTE de OMG.

En la siguiente tabla se resume la correspondencia entre los términos que se estaban utilizando en MAST 1.x, los términos que se utilizan en MARTE y los que finalmente se han adoptado en MAST 2.

MAST 1.x	MARTE	MAST 2
Processing_Resource	ProcessingResource	Processing_Resource
Processor	ComputingResource	Computing_Resource
Regular_Processor	SA_ExecHost	Regular_Processor
Network	CommunicationMedia	Network
Packet_Based_Network	SA_CommHost	Packet_Based_Network
	Device	
System_Timer	ClockResource	System_Timer
Alarm_Clock		Alarm_Clock
Ticker		Ticker
Primary_Scheduler	Scheduler	Primary_Scheduler
Secondary_Scheduler	SecondaryScheduler	Secondary_Scheduler
Scheduling_Policy	SchedulingPolicy	Scgeduling_Policy
EDF	EarliestDeadlineFirst	EDF
FP_Packet_Based		FP_Packet_Based
	TimeTableDriven	Timetable_Driven
Scheduling_Parameters	SchedulingParameters	Scheduling_Parameters
Scheduling_Server	Schedulable_Resource	Schedulable_Resource
	CommunicationChannel	Communication_Channel
Operation	ResourceUsage	Operation
Simple		Simple
Message_Transmission		Message_Transmission
Composite		Composite
Enclosing		Enclosing
Shared_Resource	MutualExclusionResource	Mutual_Exclusion_Resource
Real_Time_Situation	AnalysisContext	Real_Time_Situation
Transaction	EndToEndFlow	End_To_End_Flow
Activity	Step	Step
Event	Event	Event
External_Event	WorkloadEvent	Workload_Event
Regular_Event		Regular_Event
	TimedObserver	Observer
Timing_Requirement		Timing_Requirement
		Queue_Size_Requirement
Event_Handler	PresedenceRelantionship	Event_Handler
Concentrator	Merge	Merge
Barrier	Join	Join
Multicast	Fork	Fork
Delivery_Server	Branch	Branch
Query_Server		Query_Server

La alineación no ha sido completa porque, a veces, el término es extraño fuera del contexto de las otras secciones de MARTE; así por ejemplo, SA_ExecHost hace referencia a su especialización en la sección *Schedulability Analysis* de MARTE, y se ha optado por el término antiguo *Regular_Processor* que todo el mundo entiende. En otros casos, porque el término representaba elementos con notables diferencias conceptuales, y su adopción podía conducir a una mayor confusión.

Por último, hay que resaltar que entre MARTE y MAST hay una importante diferencia que hace a veces confuso establecer la equivalencia entre sus elementos. En MAST se utiliza la dualidad especificación/instancia, mientras que en MARTE sólo existen instancias ya que siempre modela escenarios concretos. Por ejemplo, en MAST se utiliza la dualidad *Operation*/*Step*, donde la operación describe la capacidad de procesamiento que requiere la ejecución de una sección de código (en abstracto, utilizando métricas normalizadas) y *Step* representa el uso (en tiempo físico) de un procesador en el contexto de una respuesta a un evento concreto y ejecutada por un thread específico. En MARTE sólo existe el concepto *Step*.

4 Nuevos aspectos relativos al modelado de la plataforma de ejecución.

El modelado de las plataformas de ejecución ha sido tradicionalmente uno de los puntos fuertes de MAST. Trata los procesadores (*Computing_Resources*) y las redes (*Networks*) bajo el concepto unificado de *Processing_Resource* y ello facilita la extensión de las técnicas de análisis y diseño de sistemas monoprocesadores a sistemas distribuidos. En la versión MAST 2 se han modificado tres aspectos importantes:

1. **Sincronización de los Computing_Resource y Networks de una plataforma distribuida**: A tal fin, se han introducido los elementos *Synchronization_Timer* que representan relojes globales que se utilizan como servidores de tiempo para sincronización. Cuando en el modelo de un sistema, varios *Processing_Resources* referencian a un mismo *Synchronization_Timer* significa que los relojes de todos ellos se encuentran sincronizados. El atributo *Precision* de los relojes *Synchronization_Timer* describe el máximo error entre las dataciones por los relojes sincronizados con él.

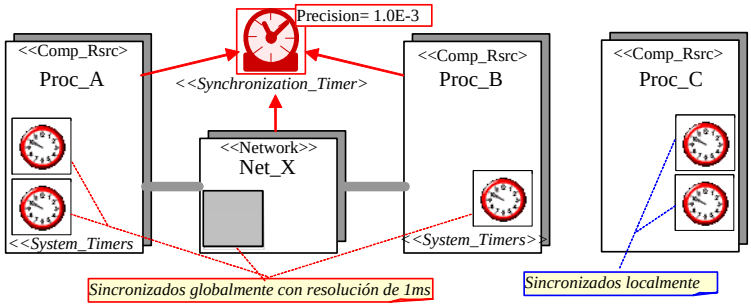

Fig. 3. Múltiples relojes en una plataforma distribuida en MAST 2

2. **Capacidad de utilizar múltiples System_Timer en un Computing_Resource**: Se permite definir múltiples *System_Timer* en un *Computing_Resource*. Para cada uno de ellos se definen sus propios consumos de gestión (*Overheads*) y sus resoluciones específicas. Los relojes de un mismo *Computing_Resource* están sincronizados entre sí, y en cada caso debe ser especificado cual de ellos es utilizado como base para los eventos (*Timed_Event*) y en los manejadores de eventos temporizados (*Delay* y *Offset*).
3. **Redes complejas *Switches* y *Routers***: Se ha incluido la capacidad de modelar plataformas distribuidas con redes complejas compuestas por múltiples *Networks* interconectados por *Switches* y *Routers*. Los modelos de estos elementos contienen la información de los retrasos y alternativas de enrutamiento que se producen en ellos cuando transmiten un mensaje que se transfiere de una a otra red. La topología de la red no se describe explícitamente en el modelo, sino que en el contexto de los *End_To_End_Flow* es donde se describen las rutas que siguen los mensajes. Tanto en los *Switches* como en los *Routers* se admite el modelado de la replicación de mensajes, y en el caso de los *Routers*, se admite también el modelado de enrutamientos alternativos.

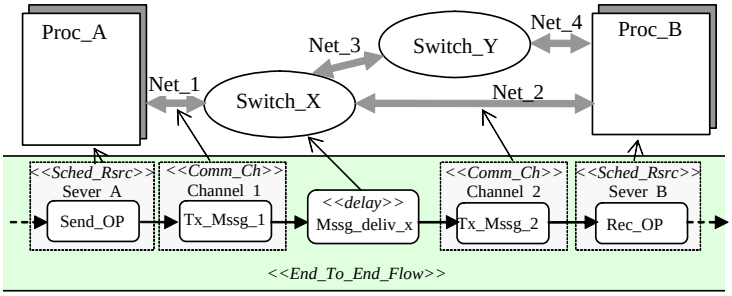

Fig. 4. Enrutamiento de un mensaje por una red compleja

5 Nuevos aspectos en planificadores y recursos planificables.

En MAST, los *Scheduler* describen las estrategias de gestión de la capacidad de procesamiento de los *Processing_Resources*, y los elementos *Schedulable_Resources* o *Communication_Channel* modelan las unidades que pueden ser planificadas por los *Schedulers* y en los que se agrupan los *Steps* que modelan la ejecución de acciones y actividades de las respuestas de la aplicación.

MAST permite modelar una estructura jerarquizada en múltiples niveles de *Schedulers*. Se denomina *Primary_Scheduler* al que se encuentra en el nivel inferior y gestiona la totalidad de la capacidad del *Processing_Resource*. Se denominan *Secondary_Scheduler* a los planificadores que gestionan una parte de la capacidad de procesamiento, la cual, ya ha sido distribuida previamente por un planificador de nivel inferior. La jerarquización de los *Scheduler* ya estaba definida en MAST, y es la base de las estrategias de planificación de aplicaciones de tiempo real basadas en componentes, o que se despliegan en plataformas abiertas.

La estrategia de gestión de cualquier *Scheduler* la define su *Scheduling_Policy.* En MAST ya estaban definidas las políticas *Fixed_Priority_Policy, EDF_Policy* y *Packet_Based_Policy.* En la versión MAST 2 se han añadido dos nuevas políticas de planificación:

1. ***Timetable_Policy***: que permite gestionar la capacidad del *Processing_Resource* en base a ventanas temporales en las que se ejecutan independientemente los diferentes *Schedulable_Resource* que se asignan a cada una (o grupo) de ellas. Estas políticas de planificación que son la base de los sistemas de tiempo real basados en separación temporal, y se pueden aplicar tanto a la gestión de la capacidad de los *Computing_Resources* como de los *Network.*
2. ***AFDX_Policy***: es la política de control de la transmisión en redes de tiempo real de tipo AFDX (*Avionics Full Duplex Switched Ethernet*). En estas redes se controla la capacidad de envío sobre la red a través de enlaces virtuales (*Virtual Links*). Esta es la política que se usará para redes conectadas a través de un switch AFDX.

Los *Scheduling_Parameters* que se asocian a los *Schedulable_Resources* modelan la forma específica en que deben ser planificados por los *Schedulers*. Obviamente, el parámetro que utiliza debe ser compatible con la política del *Scheduler* que lo planifica. En MAST ya estaban definidos los *Scheduling_Parameters*: *Non_ Preemptible_Params, Fixed_Priority_Params, Interrupt_FP_Params, Polling_ Params* y *Sporadic_Server_Params* para los planificadores con políticas basadas en prioridades (*Fixed_Priority_ Policy* y *Packet_Based_Policy*), y *EDF_Params* para los planificadores con *EDF_Policy.* En la versión MAST 2 se han definido nuevas formas en las que los *Schedulable_Resource* pueden ser planificados:

3. ***Resource_Reservation_Params***: modelan *Schedulable_Resources* que son planificados bajo contratos de reserva de recursos, y que una vez que han sido aceptados por el planificador, tienen garantizada la disponibilidad de una cierta capacidad de uso (*budget*) del recurso por cada periodo de tiempo, y con un retraso acotado de respuesta descrito como un *Deadline*. De esta familia de *Scheduling_Parameters* se han definido modelos de varias estrategias concretas (*Virtual_Periodic_Server, Virtual_Deferrable_Server, Virtual_Sporadic_Server* y *Virtual_Token_Bucket_Comm_Channel*). Todas ellas, corresponden a implementaciones disponibles y para las que se han descrito técnicas de análisis de planificabilidad.
4. ***AFDX_Vitual_Link_Params***: son los parámetros asociados a cada *Virtual Link* en una red de tipo AFDX. El *Virtual Link* limita el número máximo de bytes que se pueden enviar cada vez y el tiempo mínimo que debe transcurrir entre dos envíos.
5. ***Tabletime_Driven_Based_Params***: Permite definir la partición constituida por un conjunto de ventanas temporales en las que debe ser planificado el *Schedulable_Resource* bajo la *Timetable_Policy* que lo gestiona.

Con los planificadores definidos se pueden modelar sistemas con estructuras de planificación complejas, tales como los que se muestran en las siguientes figuras. En la figura 5 se muestran los elementos más relevantes del modelo en el caso de dos aplica-

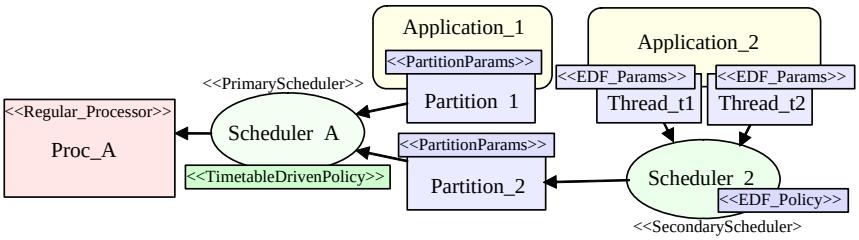

Fig. 5. Modelo MAST de un sistema soportadopor un planificador de tiempo.

ciones independientes que son planificadas en un procesador con una estrategia basada en tiempo compartido. La primera aplicación es una actividad simple periódica y es planificada directamente, la segunda es compleja y utiliza un planificador secundario con política EDF y dos threads.

En la figura 6 se muestra la estructura del modelo de la ejecución de la aplicación de tiempo real *Application_A*, sobre una plataforma distribuida y bajo el paradigma de reserva de recursos. El análisis de planificabilidad de la aplicación se puede realizar en base al conocimiento de los contratos de los tres recursos virtuales: *Server_1* y *Server_2* contratados con los dos procesadores y *Comm_Ch_N* contratado con la red de comunicación. En este caso, la aplicación se puede planificar independientemente y sin necesidad de conocer el resto de las aplicaciones que se estén ejecutando en la plataforma distribuida.

Para que la plataforma pueda aceptar los contratos de los tres recursos virtuales, necesita hacer un análisis de planificabilidad de la carga completa de la plataforma distribuida, pero en este caso no se requiere conocer Application_A (sólo los contratos de reserva de recursos que tiene definidos).

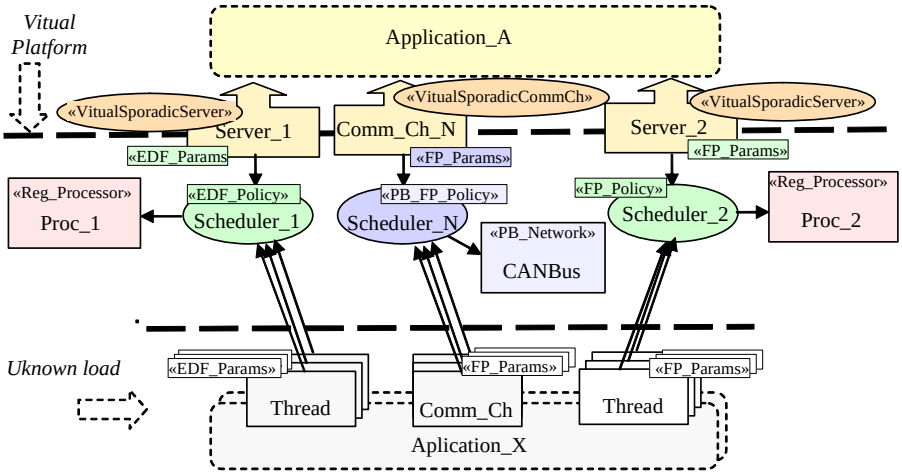

Fig. 6. Elementos del modelo de una aplicación ejecutada sobre plataforma virtual

6 Cambios en el modelo del flujo de control.

En la sección de MAST relativa al modelo del flujo de control de las actividades que constituyen la respuesta de la aplicación a los eventos externos, sólo se ha introducido un cambio relevante: la formulación explícita de la toma o liberación de los *Shedulable_Resource* en cada *Step*. En la versión inicial de MAST, se sobreentendía que cuando en el contexto de un *End_To_End_Flow* un *Step* pasaba el flujo de control a otro Step que no estaba planificado por el mismo *Shedulable_Resource*, el primero quedaba libre. Cuando se utiliza un único *Shedulable_Resource* por cada *End_To_End_ Flow* este efecto es irrelevante, sin embargo en otras situaciones más complejas, MAST no tenía capacidad de modelado.

En la figura 7, se muestra un caso en el que se requiere utilizar la extensión de MAST 2 para modelarla. El código establece que una única tarea Ada, puede ejecutar dos secciones diferentes en función de que se reciba el evento *e1* o el evento *e2*. Sin embargo, cuando se ha iniciado la respuesta a *e1*, la tarea no queda libre hasta que finalice toda ella (esto es que finalice *Oper_3*). Con el cambio introducido en MAST 2 esto queda descrito en el modelo: la tarea se toma al inicio del *Step* que ejecuta *Oper_1* (*lock=true*) y permanece tomada hasta que finaliza el *Step* que ejecuta la operación *Oper_3* (*unlock=true*). A partir de ese momento, se vuelve a iniciar el bucle y se elige la sección de código que va a ser ejecutada con la tarea.

```
....
loop
    select
          accept e1;
          Oper_1
          remote_call (Oper_2);
          Oper_3
    or
          accept e2;
          ....
    end select;
end loop;
```

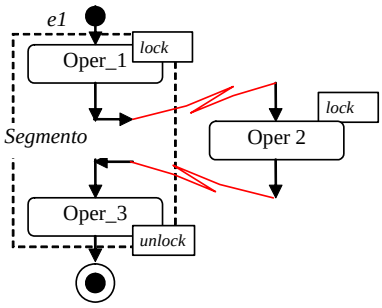

Fig. 7. Formulación explícita de la toma y liberación de un *Schedulable_Resource*

7 Requisitos no funcionales.

En la versión previa de MAST se incorporaban los requisitos temporales ya que constituyen una información que se necesita para realizar los análisis de planificabilidad de los sistemas modelados. Los requisitos temporales se definen en el contexto de los *End_To_End_Flow* y se formulaban asociándolos a la ocurrencia de los eventos internos, que representan los estados de ejecución de las respuestas.

En la versión MAST 2, se extiende la noción de requisito a aspectos no funcionales de cualquier naturaleza. Siguiendo la nomenclatura de la especificación MARTE, a los eventos se les puede asociar opcionalmente un elemento de tipo *Observer* que representa un requisito no funcional. En la versión actual, se tienen definidos dos tipos con-

cretos de *Observer*: los requisitos temporales (*Timing_Requirement*), y *Queue_Size_Requirement* que corrresponde a requisitos relativos al número de instancias de eventos que pueden encolarse en un punto de ejecución del *End_To_End_Flow*. Este tipo de requerimiento se utiliza para diseñar los sistemas de forma que quede garantizado que en el peor caso nunca se sobrepasa la capacidad de las colas que almacenan los datos asociados a los eventos pendientes.

8 Conclusiones y calendario.

En este momento, se ha formulado el metamodelo de la versión MAST 2. Aún falta una gran cantidad de trabajo hasta que se consiga que todo el entorno MAST, y con él las herramientas de análisis y diseño existentes, esté plenamente operativo en la nueva versión. Este proceso de adaptación requiere un gran esfuerzo, por lo que se prevé que no esté concluido antes de que finalice el año actual 2010.

Sin embargo, consideramos importante dar una gran difusión a la nueva versión de MAST 2, ya que la etapa de redefinición del modelo es el momento adecuado para introducir las grandes decisiones de estrategias de modelado, y pedimos que aquellos grupos que consideren que MAST puede constituir una plataforma de prueba y desarrollo de sus futuros trabajos, analicen las opciones que se están adoptando, y en su caso, propongan nuevas alternativas.

References

[1] González Harbour M., Gutiérrez García J.J., Palencia J.C., y Drake J.M.: "MAST: Modeling and Analysis Suit for Real Time Applications" Proceeding of the 13th Euromicro Conference on Real Time Systems. Pp.125-134, Delft (Netherlands), June, 2001.P. López, J.M. Drake, P. Pacheco, and J.L. Medina: An Ada 2005 Technology for Distributed and Real-Time Component-based Applications. Proc. 13th Intl. Conf. on Reliable Software Technologies Ada-Europe, Venice, 2008.

[2] Object Management Group, UML Profile for Schedulability, Performance, and Time Specification, Version 1.1, OMG doc. formal/05-01-02, 2005.IST project COMPARE: Component-based approach for real-time and embedded systems http://www.ist-compare.org

[3] Object Management Group, UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) RFP, OMG doc. ptc/07-08-04, 2007.