Revista Iberoamericana de Automática e Informática Industrial
CEA IFAC comité español de automática
RIAI
ISSN: 1697-7912. Vol. 3, Núm. 2, Abril 2006, pp. 28-39
http://riai.isa.upv.es

EN BUSCA DE LA INTEGRACIÓN DE HERRAMIENTAS DE TIEMPO REAL A TRAVÉS DE UN MODELO ABIERTO

J.M. Drake, M. González Harbour, J.J. Gutiérrez, J.L. Medina y J.C. Palencia

Grupo de Computadores y Tiempo Real. Universidad de Cantabria

Resumen: Se presenta el entorno MAST, que integra herramientas de análisis y diseño de sistemas distribuidos de tiempo real, facilitando su interoperatividad a lo largo del proceso de desarrollo, y está abierto para extender su capacidad de modelado e implementar herramientas con nuevos algoritmos o métodos de diseño. Se describe su arquitectura, basada en la especificación formal, estandarizada y extensible de las estructuras de datos que soportan los modelos de tiempo real y los resultados de las herramientas. Estas estructuras de datos y la metodología de modelado constituyen la base de la interoperatividad entre las herramientas que incorpora. Se presentan las herramientas disponibles y se justifica su adecuación para el desarrollo de aplicaciones distribuidas de tiempo real desarrolladas con Ada 95 o sobre sistemas operativos de tiempo real con interfaz POSIX 1003.13. *Copyright © 2006 CEA-IFAC*

Palabras Clave: Tiempo real, sistemas distribuidos, herramientas CASE, análisis de planificabilidad.

1. PRESENTACIÓN Y ANTECEDENTES

Los sistemas de tiempo real que actualmente se utilizan en la industria (automovilística, aeroespacial, de telecomunicación, de electrodomésticos, etc.) van incrementando ostensiblemente su complejidad. Utilizan plataformas computarizadas distribuidas constituidas por decenas de nudos procesadores, que alojan diferentes aplicaciones independientes o acopladas entre sí, muchas de las cuales tienen requisitos de tiempo real y niveles preestablecidos de calidad de servicio. El incremento de la complejidad del software de estos sistemas distribuidos y de tiempo real ha estado acompañado de la correspondiente generación de herramientas y recursos que hacen posible su desarrollo, así:

- Actualmente, se dispone de sistemas operativos de tiempo real escalables y con interfaces estandarizadas (IEEE, 2003) que permiten estrategias de planificación flexible y adaptativa, y que hacen posible redistribuir eficientemente los recursos para satisfacer tanto los requerimientos temporales estrictos y laxos, como los niveles requeridos de calidad de servicio.
- Se han propuesto diferentes metodologías de diseño (Gomaa, 2000; Kabous, 2002; Douglass, 2000) que definen procesos de desarrollo que garantizan los niveles de calidad del software, y hacen previsible el costo y el esfuerzo que requiere su desarrollo.
- Se han desarrollado métodos de análisis de planificabilidad, de despliegue y de asignación de prioridades que son muy eficientes en el análisis, el diseño y la verificación de los sistemas de tiempo real (Klein, *et al.*, 1993; Liu, 2000; Cheng, 2002).
- Por último, se han propuesto (Gopalan, 2001; Deng y Liu, 1997) nuevos servicios en los sistemas operativos de tiempo real destinados a compartir los recursos en el tiempo, o distribuir franjas de anchura de banda de procesadores y redes de comunicación entre las tareas, que hacen posible la planificación cuando la carga cambia en el tiempo.

Aunque todos estos recursos están descritos en la bibliografía, el diseñador industrial tiene problemas para su aplicación tanto por la complejidad de sus algoritmos, como por la cantidad de información que debe ser procesada a lo largo de las sucesivas etapas del ciclo de desarrollo. Para que el diseñador pueda aplicar todos estos recursos, necesita disponer de entornos CASE amigables en los que se ofrezcan las diferentes herramientas que los implementan y se facilite el intercambio de modelos y resultados entre ellas.

Una característica que hace que el diseño de los sistemas de tiempos real sea mas complejo que el de otros tipos de software, es que el tiempo no es una magnitud que se gestiona explícitamente en los lenguajes de programación, ni puede habitualmente ser establecido en el acceso a los servicios de los sistemas operativos. Por ello, en el diseño de los sistemas de tiempo real se tiene que superponer a la descripción funcional convencional, una nueva descripción o modelo del comportamiento temporal que contemple todos aquellos aspectos que son necesarios para evaluar la temporalidad de las respuestas. Este modelo de tiempo real es también el ámbito en el que se negocian los requerimientos de respuesta temporal en las fases de especificación del sistema, se razona sobre su estructura en las fases de análisis y diseño y se certifica el cumplimiento de los requerimientos temporales en las fases de validación. Hasta ahora, han sido propuestas diferentes estrategias de modelado de los sistemas de tiempo real (Kopetz, *et al.*, 1991; Selic, *et al.*, 1994; OMG, 2003), estando todas ellas íntimamente relacionadas con los diferentes métodos de análisis que existen y las herramientas que los utilizan.

Nuestro Grupo[1] ha desarrollado durante los últimos años el entorno MAST[2] (*Modeling and Analysis Suite for Real-Time Applications*) que constituye una plataforma CASE abierta cuya finalidad es servir de base de integración y de interoperatividad para herramientas de análisis y diseño de sistemas de tiempo real.

En la figura 1 se muestran los usuarios potenciales y las formas de utilización previstas del entorno MAST:

- Al *investigador de tiempo real* le ofrece una metodología de modelado abierta y extensible que le permite incorporar los nuevos aspectos del sistema de tiempo real que esté ideando, así como un entorno amigable en el que desarrollar y validar la nuevas técnicas y herramientas en que trabaja.

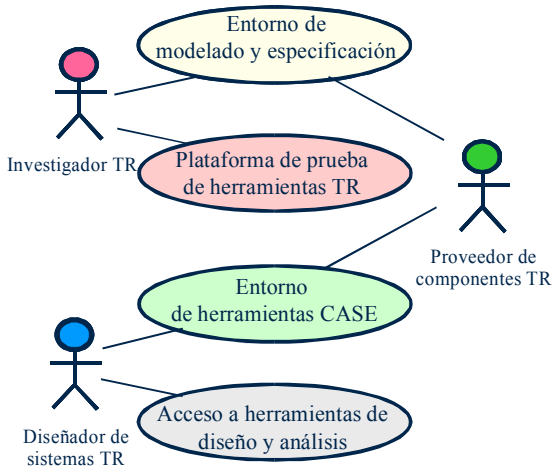

Figura 1. Usuarios y Casos de Uso del entorno MAST.

- Al *diseñador industrial* de sistemas de tiempo real le ofrece un entorno en el que, sobre un modelo de tiempo real único y próximo a la concepción que tiene del sistema que desarrolla, se dispone de un abanico de herramientas que puede aplicar sin necesidad de conocer los algoritmos en que se basan.
- Al *proveedor de herramientas y componentes* de tiempo real le ofrece una metodología de modelado con la que describir sus productos (sistema operativo, middleware, componentes software, etc.).

2. ENTORNO ABIERTO MAST

Los dos aspectos que constituyen la base del entorno MAST son la metodología de modelado elegida y la especificación de la estructura de datos con que se representan los modelos y los resultados que generan las herramientas. Como se muestra en la figura 2, las herramientas que se integran en el entorno lo hacen a través de la estructura de datos.

El modelo de tiempo real de un sistema es una representación abstracta que describe las características necesarias para poder predecir el comportamiento temporal del sistema. Obviamente, no existe una única metodología de modelado, sino que con cada metodología de diseño e incluso con cada técnica de análisis que se vaya a aplicar suele proponerse una variante nueva.

Afortunadamente, la tecnología con la que se construyen los sistemas de tiempo real está actualmente bastante consolidada y se basa en sistemas operativos que soportan concurrencia a través de hilos, están dotados de planificadores de respuesta predecible y mecanismos de sincronizaciónque son compatibles con las técnicas de análisis de tiempo real. La metodología de modelado que se ha elegido se basa en conceptos que son propios de esta tecnología de implementación.

[1] Grupo de Computadores y Tiempo Real de la Universidad de Cantabria. http://www.ctr.unican.es

[2]Modeling and Analysis Suite for Real-Time Aplications (versión 1.3.6). http://mast.unican.es

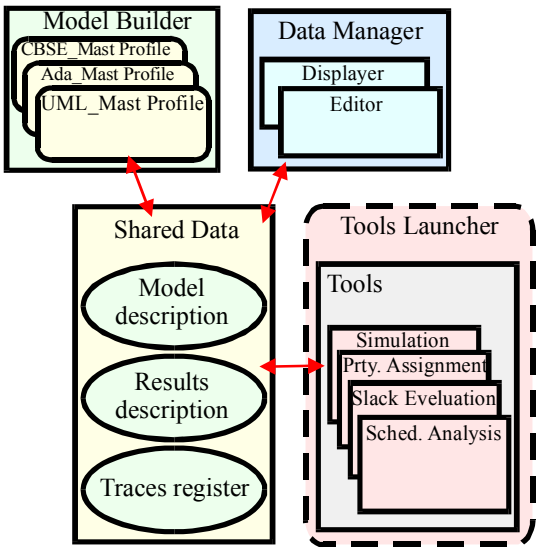

Figura 2. Estructura del entorno MAST.

Aunque esta tecnología es actualmente la más utilizada, sobre todo en sistemas complejos, es importante resaltar que hay otras tecnologías que no se pueden incluir en este paradigma y a ellas no es extensible la metodología de modelado que se propone en el entorno MAST. Ejemplos de estos casos son los sistemas basados en planificadores cíclicos o los conducidos por tablas de tiempos de activación (*Time triggered*). Otros paradigmas que resultan muy difíciles de modelar con MAST son los basados en máquinas virtuales que ejecutan código generado automáticamente a partir de descripciones basadas en máquinas de estados (Gullekson, 2000).

El entorno MAST se ha diseñado buscando que ofrezca características de sistema abierto, y permita que cualquier usuario pueda extender su capacidad de modelado con nuevos conceptos. Con tal fin, se ha utilizado una estrategia del tipo orientada a objetos en la definición de los elementos de modelado. Todos ellos se derivan de un reducido conjunto de clases abstractas que representan los conceptos básicos y comunes, que cualquier diseñador utiliza para concebir el sistema de tiempo real que desarrolla. Estas clases raíces se especializan posteriormente en nuevas clases concretas que modelan conceptos propios de la tecnología de tiempo real específica que se utiliza.

La definición, estable y documentada de las estructuras de datos es la base de la interoperatividad de las herramientas y de que el entorno esté abierto a que el usuario pueda incorporar otras nuevas. Las estructuras de datos están codificadas en ficheros de texto con dos formatos: el original que es conciso y está soportado por librerías que ofrece el propio entorno para su gestión, y otro formulado en XML y formalizado mediante plantillas *Schema*, que puede ser validado, interpretado y procesado utilizando los recursos que proporciona la tecnología XML.

Como se muestra en la figura 2, en el entorno se pueden identificar tres tipos de herramientas:

- *Herramientas de gestión del modelo y de los resultados*: compuestas por interfaces gráficas para la introducción, edición y visualización de los modelos y de los resultados y trazas generadas por las herramientas.
- *Herramientas de análisis y diseño de tiempo real*: que procesan el modelo de tiempo real del sistema y generan información útil para el análisis y diseño del sistema que se desarrolla.
- *Herramientas de generación del modelo*: que ayudan a generar el modelo de tiempo real a partir de la especificación, de la descripción funcional, lógica y de despliegue o del código que se generan en el proceso de desarrollo.

3. METODOLOGÍA DE MODELADO

En MAST se ha adoptado la metodología de modelado denominada transaccional, que ha sido utilizada por muchos autores desde de que la teoría de planificabilidad por ritmo monotónico (RMA) (Klein, *et al.*, 1993) se comenzó a utilizar. Este modelo resulta de dos descripciones superpuestas: la descripción reactiva del sistema como una secuencia de actividades que se ejecutan concurrentemente como respuesta a los eventos externos o temporizados que se producen en el sistema, y la sincronización que debe establecerse entre la ejecución de las actividades como consecuencia de que han de utilizar recursos que requieren ser usados en régimen de exclusión mutua.

Se propone la construcción de un modelo por cada situación de tiempo real (*RealTtime Situation*). Con ella se representa un modo de operación del sistema con una determinada carga de trabajo, la cual se describe mediante un patrón de generación de eventos externos, y un conjunto de requerimientos temporales relativos a la ejecución de las actividades que conlleva la respuesta a esos eventos. La situación de tiempo real constituye el ámbito sobre el que operan las herramientas de análisis y diseño. El objetivo del diseño de tiempo real consiste en garantizar que cada una de las situaciones de tiempo real que puede alcanzar satisfaga los requerimientos de tiempo real estrictos y laxos y los niveles de calidad de servicio establecidos en su especificación.

Una situación de tiempo real se modela como un conjunto de transacciones de tiempo real (*Transactions*) cuya ejecución concurrente constituye la funcionalidad del sistema. Cada transacción representa la ejecución de una secuencia de actividades (*Activities*) relacionadas entre sí mediante relaciones de flujo, que constituyen la respuesta del sistema a un conjunto de señales externas o de temporización que denominamos eventos externos (*External_Events*).

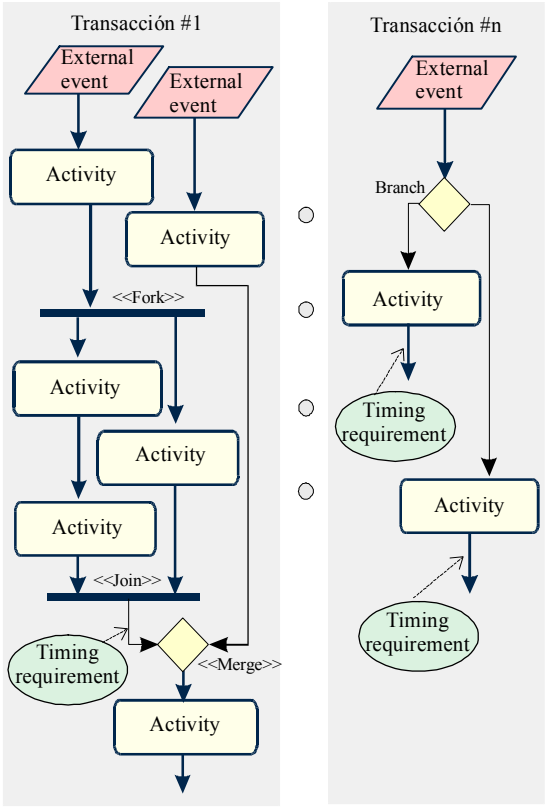

Figura 3. Modelo de transacciones de una situación de tiempo real.

Como ilustra la figura 3, la transacción constituye además el marco dentro del que se definen los requisitos temporales (*Timing_Requirements*), que se asocian a los eventos internos que representan el flujo de control dentro de la transacción.

Las actividades representan la ejecución de una operación (*Operation*) dentro del ámbito de una transacción. La operación describe una acción del sistema y está caracterizada por su temporización y por los recursos que requiere su ejecución. Cada operación representa una misma acción (ejecución de un segmento de código, transferencia de un mensaje, o la actividad de un dispositivo hardware) y puede ser llevada a cabo como parte de la respuesta de diferentes transacciones o en diferentes fases de una misma transacción.

Como ilustra la figura 4, la actividad lleva asociado el recurso de procesamiento (*Processing_Resource*) concreto que la ejecuta, y en su caso, cada operación describe los recursos compartidos (*Shared_Resource*) que necesita para su ejecución. En el caso habitual, de que varias actividades deban ser ejecutadas por un mismo recurso de procesamiento, se declara para cada actividad el servidor de planificación (*Scheduling_Server*) dentro del que se ejecuta. El servidor de planificación representa un hilo de concurrencia y tiene asociada la política de planificación (*Scheduling_Policy*) con la que el planificador del recurso (*Scheduler*) controla su ejecución.

Todos los componentes raíz, de tipo abstracto que constituyen la base de la metodología de modelado del entorno MAST, como *Transaction, ,Activity, Operation, External_Event, Timing_Requirement, Event_Handler, Processing_Resource, Scheduler, Scheduling_Server y Shared_Resource*, deben ser especializados en componentes concretos una vez que se aborde el modelado dentro de una tecnología específica. Esta extensión requiere asignar al componente concreto que se define nuevos atributos que describen características del comportamiento que modelan y nuevas posibilidades de asociación con otros elementos del modelo.

Los recursos de modelado que tiene ya definidos actualmente MAST permiten modelar aplicaciones de tiempo real desarrolladas utilizando un sistema operativo de tiempo real que ofrezca sus servicios de acuerdo con el estándar POSIX (IEEE, 2003) o que sean desarrollados utilizando un lenguaje de programación Ada que sea conforme con las extensiones de tiempo real y de sistema distribuido definidas en los anexos D y E del Manual de Referencia del lenguaje Ada (ISO, 1995).

Las transacciones tienen definidas una gran variedad de elementos (*Event_Handler*) para el control del flujo de eventos entre las actividades que las componen. Así, admiten relaciones de bifurcación (*branch*), convergencia (*merge*), diversificación y sincronización de líneas de flujo (*fork* y *join*), y también elementos para la suspension temporizada (*delay* y *offset*).

El conjunto de patrones de generación de eventos externos que se han definido, permite modelar las principales cargas de trabajo que se suelen presentar en las aplicaciones industriales. Para cada clase concreta, se definen los parámetros que caracterizan el comportamiento que con ella se describe.

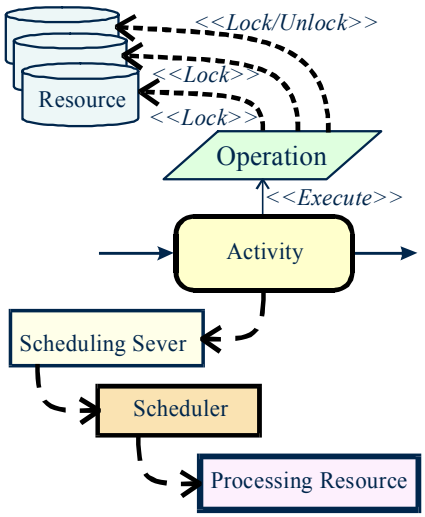

Figura 4. Recursos que relaciona una Actividad.

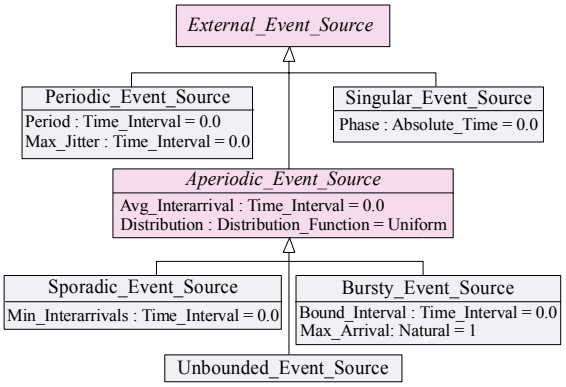

Figura 5. Patrones de generación de eventos externos.

En la figura 5 se muestra el conjunto de patrones de generación de eventos externos que se han definido.

Los requerimientos temporales representan las restricciones temporales que se han impuesto sobre la ocurrencia de un evento de flujo (finalización de una actividad) dentro del marco de referencia que constituye una transacción. Normalmente estos requisitos se establecen en relación a algún evento perteneciente o relacionado con la transacción. Un plazo (*Deadline*) establece el máximo tiempo permitido para que se concluya una actividad de la transacción. El plazo es local si hace referencia al inicio de la propia actividad, y es global si hace referencia a un evento externo. Asimismo se definen otros tipos de requerimientos temporales, unos que delimitan las fluctuaciones en los tiempos de respuesta (*Jitter*) y otros que determinan las tasas de fallos que son admisibles (*Max_Miss_Ratio*). La figura 6 muestra los tipos de requerimientos temporales que están definidos en la versión actual.

Un componente de modelado del tipo abstracto raíz *Processing_Resource* modela la capacidad de procesamiento de un recurso hardware que lleva a cabo actividades del sistema. En la figura 7, se muestran los componentes de modelado derivados de él que se han definido en la versión actual. En el primer nivel se han definido dos tipos abstractos muy generales, el procesador (*Processor*) que describe un recurso que lleva a cabo actividades consistentes en la ejecución de segmentos de código, y la red de comunicaciones (*Network*) que describe un recurso que lleva a cabo actividades que consisten en la transferencia de mensajes entre procesadores. De cada uno de ellos se ha definido un componente concreto.

El procesador estándar (*Regular_Processor*) modela los aspectos básicos del comportamiento que se derivan del hardware. Describe su capacidad relativa de procesamiento (*Speed_Factor*) y aspectos relativos a la atención de interrupciones hardware, como son las cargas de procesamiento que implican su atención y los rangos de prioridades con las que son gestionadas.

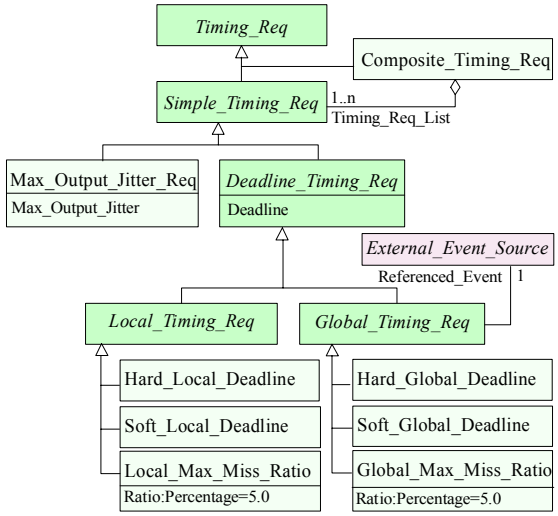

Figura 6. Tipos de requerimientos temporales.

El componente temporizador del sistema (*Timer*) modela el dispositivo hardware de que dispone el procesador para controlar los eventos temporizados y es un recurso muy relevante en un sistema de tiempo real. Este elemento describe tanto la granularidad temporal con la que el sistema es capaz de definir los eventos temporizados, como la carga de trabajo que su gestión requiere del procesador. Actualmente se tienen definidos dos tipos de temporizadores concretos: el *Alarm_Clock*, que modela dispositivos que generan los eventos temporizados mediante registro previo, y el *Ticker* que modela dispositivos que gestiona el tiempo mediante interrupciones periódicas.

La red de comunicaciones basada en paquetes (*Packet_Based_Network*) modela una red que utiliza alguna clase de protocolo basado en la transferencia de los mensajes como secuencias de paquetes, cada uno de los cuales constituye un uso no interrumpible de la red. Los atributos asociados a este tipo concreto describen su anchura de banda, su naturaleza (*Simplex*, *Half_Duplex* o *Full_Duplex*), el tamaño de paquete, y el tiempo que requiere la transferencia de un paquete.

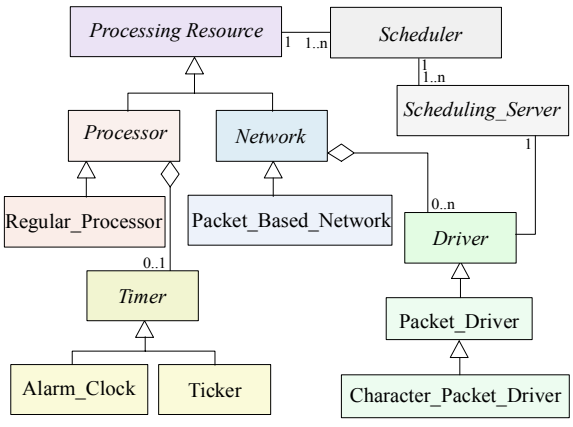

Figura 7. Componentes de modelado relacionados con los Processing_Resource.

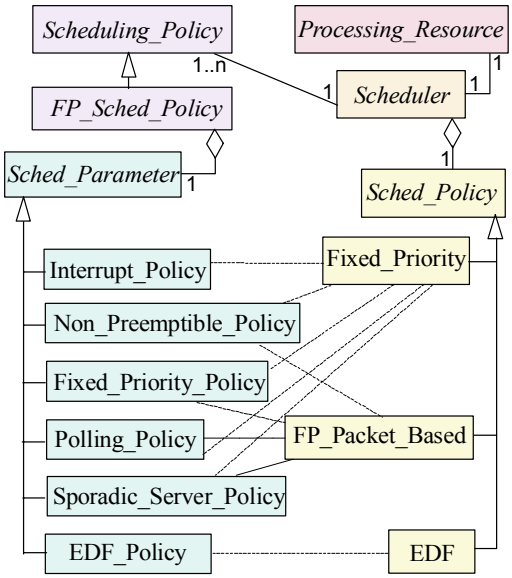

Figura 8. Tipos de planificadores y servidores de planificación.

Asociados a las redes y también a los procesadores, se definen los *Drivers*, que modelan la carga de trabajo que requiere del procesador (que envía o recibe) la transferencia de un paquete por la red, y que depende del tipo de protocolo que se utiliza.

Los planificadores (*Scheduler*) modelan las operaciones de planificación de actividades que lleva a cabo el sistema operativo. Tiene definidos atributos que describen dos características de su planificación: la estrategia de planificación (*Policy*) que lleva a cabo y la carga de trabajo que el algoritmo de planificación impone sobre el procesador. En la figura 8, se indican los tres tipos de políticas de planificación que se tienen definidas (*Fixed Priority*, *Earliest Deadline First* y *Fixed Priority Based Packet*), así como los parámetros de planificación compatibles con ellas.

Los servidores de planificación (*Scheduling_Server*) modelan los entes de planificación en que se agrupan las actividades a fin de gestionar su planificación y se corresponden con los procesos o hilos de flujo (*threads*) en el caso de los procesadores o con canales virtuales de comunicación en el caso de las redes. Están caracterizados por ciertos atributos (*Scheduling_Parameters*) que son función del tipo de planificación que se requiere, pero que deben ser compatibles con el planificador que los gestiona.

Los recursos compartidos (*Shared_Resources*) modelan los mecanismos de sincronización que proporciona el sistema operativo, o el lenguaje, a fin de desarrollar una programación concurrente segura. Ejemplos de estos mecanismos son las secciones críticas, los objetos protegidos, los mutexes, etc. Son elementos muy relevantes en los sistemas de tiempo real ya que son causa de contención en la ejecución de una respuesta, y tienen definidos protocolos específicos de acceso a fin de que no se introduzcan

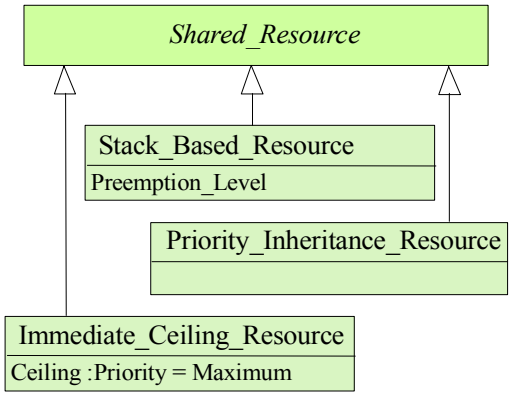

Figura 9. Recursos Compartidos pasivos.

inversiones de prioridad en la ejecución de los procesos que se encuentran a la espera de acceder a ellos.

Como se indica en la figura 9, MAST, dispone de tres tipos concretos de recursos pasivos: el recurso del tipo techo de prioridad (*Inmediate_Ceiling _Resource*) que implementa el protocolo *Priority Ceiling* de Ada o su equivalente *Priority Protect* de POSIX y el recurso de tipo de herencia de prioridad (*Priority_Inheritance_Resource*) que sigue el protocolo de herencia prioridad básico, son propios de sistemas con planificador de prioridades fijas, mientras que el SRP (*Stack_Based_Resource*) que sigue el protocolo Stack Based Protocol (*SRP*) el cual permite sincronizar actividades que se planifican por prioridades fijas, con otras planificadas por EDF.

El componente de modelado operación (*Operation*) describe una actividad que es realizada en alguna fase de la ejecución del sistema y que o bien requiere un tiempo su ejecución o requiere acceder a recursos para ser ejecutada. Ejemplos de operaciones son la ejecución de un código por un procesador, la actividad de un dispositivo hardware o la transferencia de un mensaje a través de una red de comunicaciones.

La operación es el único componente que tiene las características de descriptor, esto es, su declaración dentro de un modelo sólo representa la descripción de una tarea definida en el sistema y no su ejecución. Es justamente mediante la declaración de una actividad que la tenga asociada que se establece en qué etapa de una transacción se lleva a cabo. Se definen tres tipos de operaciones.

Una operación simple (*Simple_Operation*) es una actividad secuencial que se ejecuta en su totalidad dentro de un servidor de planificación y que sólo admite sincronización con alguna otra actividad en su inicio cuando requiere los recursos que necesita para ser llevada a cabo. Sus atributos principales son la capacidad de procesamiento que requiere y la lista de recursos a los que debe acceder en su inicio y los que liberará con su finalización.

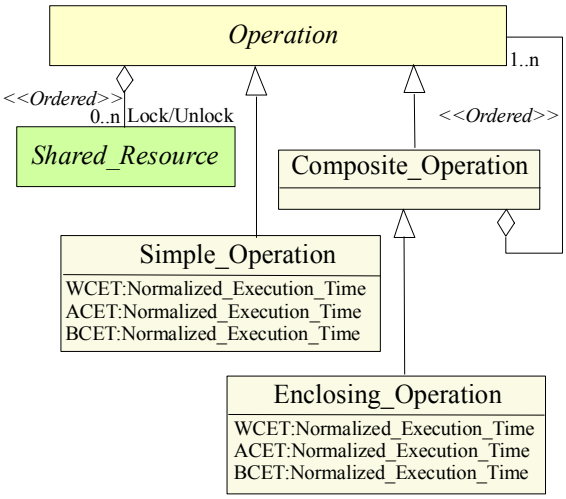

Figura 10. Tipos de operaciones definidas.

La capacidad que requiere se especifica de forma probabilística (peor, mejor y caso promedio), y se formula mediante tiempos normalizados respecto a un recurso de procesamiento con capacidad relativa igual a 1.

Una operación compuesta (*Composite_Operation*) describe la ejecución de una secuencia de operaciones (simples o compuestas) previamente definidas. En este caso los parámetros de capacidad de procesamiento son los de las operaciones que contiene, y el atributo principal es justamente la lista ordenada de operaciones de que se compone.

Por último, las operaciones englobantes (*Enclosing_ Operation*) son operaciones compuestas en las que, por su naturaleza, es más sencillo estimar su temporización de forma global que medir la parcial de cada operación simple de que se compone. En estas operaciones se incluyen atributos que describen la capacidad de procesamiento global de la operación. Sin embargo, la información relativa a las operaciones de que se compone es necesaria si éstas especifican recursos compartidos. La figura 10 muestra los tres tipos de operaciones que están definidas

La información completa y detallada de los elementos de modelado definidos en MAST, sus atributos, y relaciones se puede encontrar en la URL http://mast.unican.es.

Por otra parte, el modelado de tiempo real de un sistema es un trabajo complejo y delicado que requiere un conocimiento detallado de tres elementos:

- Modelo de la plataforma, compuesto por los modelos de todos los elementos hardware y software que soportan a la aplicación que se desarrolla, pero que no han sido diseñados específicamente para ella. Este modelo incluye los modelos de la plataforma hardware (procesadores y redes de comunicación) y los modelos del software de base (sistema operativo y drivers).
- Modelo de los componentes software específicos de la aplicación que se desarrolla, compuesto por los modelos de las operaciones que describen el comportamiento temporal de los procedimientos y funciones que son invocadas en las transacciones de tiempo real, los mensajes que se transfieren por la red y los recursos compartidos pasivos que se instancian en la aplicación.
- Modelos de las situaciones de tiempo real, que describen los conjuntos de transacciones con requerimientos de tiempo real, o aquellas cuyas actividades afectan a otras transacciones que sí son de tiempo real.

El trabajo de generación del modelo de tiempo real es muy costoso, pero también necesario. Sin él, no se puede realizar el diseño ni asegurar que su operación es correcta. La metodología de modelado MAST reduce considerablemente el trabajo de construcción del modelo de una aplicación porque al modelar independientemente la plataforma, los componentes lógicos y las situaciones de tiempo real, se consigue una mayor reusabilidad. Si una plataforma hardware, un sistema operativo o el software de comunicaciones se reutilizan en sucesivas aplicaciones, sus modelos son directamente transferibles de una a otra.

Asimismo, si los componentes software y las librerías de funciones se reutilizan, los modelos son de nuevo directamente re-usables. Cuando un grupo o empresa ha desarrollado sucesivos proyectos, tiene un alto porcentaje de los recursos ya modelados, y la construcción del modelo de la aplicación se reduce únicamente al modelado de las situaciones de tiempo real.

4. HERRAMIENTAS EN MAST

El entorno MAST contiene diversas herramientas, de análisis de planificabilidad, de cálculo de tiempos de bloqueo, y de asignación de parámetros de planificación. Estas herramientas aplican diferentes algoritmos sobre el modelo y aportan sus resultados. Algunas herramientas pueden utilizar otras en su ejecución, así por ejemplo la asignación de prioridades puede requerir el uso del análisis de planificabilidad. El diseño de estas herramientas permite optar por realizar solamente la validación de la consistencia del modelo a fin de comprobar si es correcto, o bien aplicar algunas de las técnicas de análisis de planificabilidad o de asignación de prioridades con las opciones de usuario que se hayan elegido, tales como el cálculo de los techos de los recursos, o el de la holgura (*slack*).

El esquema general del funcionamiento de estas herramientas, así como las diferentes opciones

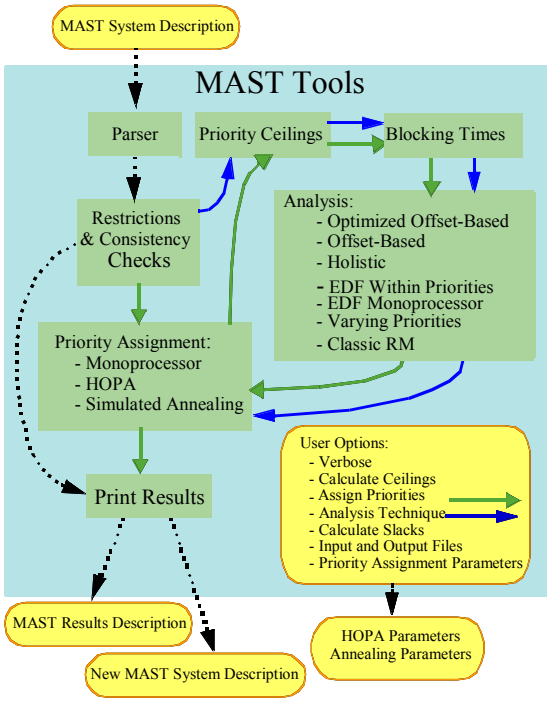

Figura 11. Esquema de utilización de las herramientas de análisis y diseño en el entorno MAST.

descritas para su utilización y las interacciones entre sus elementos internos se representan en la figura 11.

Las técnicas de análisis de planificabilidad son aplicables tanto a sistemas monoprocesadores como distribuidos, planificados por políticas basadas en prioridades fijas, EDF, y EDF sobre prioridades fijas. Estas técnicas de análisis son las siguientes:

- *Classic RM Analysis*. Análisis exacto de tiempo de respuesta para monoprocesador con prioridades fijas, desarrollado en (Harter, 1984) y (Joseph y Pandya, 1986), y más tarde extendido en (Lehoczky, 1990) para manejar plazos arbitrarios y en (Tindell, 1994b) para manejar el *jitter*. Se corresponde con la Técnica 5 de (Klein, *et al.,* 1993).
- *Varying Priorities Analysis*. Análisis de tiempo de respuesta para monoprocesador con prioridades fijas en el que las tareas pueden cambiar explícitamente sus prioridades. Desarrollado en (González, *et al.,* 1994), se corresponde con la Técnica 6 en (Klein, *et al.,* 1993).
- *EDF Monopocessor Analysis*. Análisis exacto de tiempo de respuesta para mono-procesador con EDF, desarrollado primero en (Spuri, 1996a). En la implementación MAST utilizamos *EDF Within Priorities* porque puede haber rutinas de servicio de interrupción (modeladas con priorides fijas) además de las tareas EDF.
- *EDF Within Priorities Analysis*. Mezcla del análisis de tiempo de respuesta para prioridades fijas (Klein, *et al.*, 1993; Lehoczky, 1990; Tindell, 1994b) y para EDF (Spuri, 1996a). Permite analizar sistemas con planificadores jerárquicos en los que el planificador primario se basa en prioridades fijas, y puede haber otros planificadores secundarios EDF sobre un nivel de prioridad dado. La técnica fue desarrollada en (González y Palencia, 2003).
- *Holistic Analysis*. Este análisis extiende el análisis de tiempo de respuesta a sistemas multiprocesadores y distribuidos. Fue desarrollado para sistemas planificados por prioridades fijas en (Tindell y Clark, 1994) y refinado en (Palencia, *et al*., 1997). Más tarde en (Spuri, 1996b) se extendió a sistemas planificados con EDF. No es una técnica exacta, porque asume que las tareas en la misma transacción son independientes.
- *Offset Based Analysis*. Análisis de tiempo de respuesta para sistemas multiprocesadores y distribuidos que mejora enormemente el pesimismo del *Holistic Analysis* teniendo en cuenta que las tareas de la misma transacción no son independientes a través del uso de *offsets*. Esta técnica fue introducida en (Tindell, 1994a) y extendida a sistemas distribuidos en (Palencia y González, 1998). Más tarde, fue extendida también a sistemas EDF en (Palencia y González, 2003). Aunque obtiene resultados mucho mejores que *Holistic Analysis*, tampoco es un método exacto.
- *Optimized Offset Based Analysis*. Esta es una mejora de la técnica anterior para sistemas planificados por prioridades fijas en los que las prioridades de las tareas de una transacción dada se usan junto con las relaciones de precedencia de estas tareas para obtener una estimación más precisa de los tiempos de respuesta. Desarrollada en (Palencia y González, 1999), fue más tarde mejorada en (Redell, 2003).

En MAST se calculan automáticamente los tiempos de bloqueo causados por la sincronización por exclusión mutua para cualquiera de los protocolos soportados: el de techo inmediato (Baker, 1991), herencia de prioridad (Sha, *et al*., 1990), y SRP (*Stack Resource Protocol*) (Baker, 1991).

También se permite la combinación de estos protocolos y su uso en sistemas multiprocesadores (Rajkumar, *et al*., 1988; Rajkumar, 1990), con algunas restricciones que se describen a continuación:

- Para el protocolo de herencia de prioridad básico, en (Rodríguez y García, 2003) se mostró que la mayoría de las implementaciones no siguen estrictamente las reglas (Sha, *et al*., 1990), y que por tanto el bloqueo es normalmente mayor que el que predice la teoría. Esto se tiene en cuenta en MAST cuando se calculan los términos de bloqueo para este protocolo.
- El cálculo de los bloqueos para el protococlo SRP que se utiliza para sincronización en sistemas planificados por EDF no está resuelto para el caso distribuido.

En MAST también se incluyen herramientas que realizan la asignación automática de prioridades y otros parámetros de planificación. La asignación de prioridades se realiza tanto para sistemas mono-procesadores como para sistemas distribuidos:

- En sistemas monoprocesadores, si los plazos son iguales o menores que los periodos se utiliza la asignación *Deadline Monotonic* (Leung y Layland, 1982). La asignación clásica *Rate Monotonic* (Liu y Layland, 1973) para plazos iguales a los periodos es un caso particular de la anterior. Si los plazos son superiores a los periodos se utiliza la asignación óptima propuesta en (Audsley, 1991). Esta técnica se basa en el uso iterativo del análisis de planificabilidad para diferentes soluciones hasta que se alcanza una solución planificable.
- En sistemas multiprocesadores y distribuidos el problema de asignar prioridades es más complejo, puesto que hay interrelaciones muy fuertes entre los tiempos de respuesta de diferentes recursos. Se implementan dos soluciones heurísticas basadas en la aplicación iterativa del análisis de planificabilidad. La primera se basa en el uso de la ténica de optimización *Simulated Annealing* para la asignación de prioridades, y se debe a (Tindell, *et al*., 1992). La segunda, que obtiene mejores resultados y de manera más rápida, es el algoritmo HOPA desarrollado en (Gutiérrez y González, 1995).

Finalmente, con el conjunto de herramientas incluido en MAST no sólo se puede determinar si el sistema es planificable, sino también cuán lejos está de serlo, o cuánto le sobra y se podría utilizar sin que dejara de serlo. Esto se hace mediante el cálculo de holguras (*slacks*), que se definen como el porcentaje en que se puede incrementar el tiempo de ejecución de algunas operaciones manteniendo el sistema planificable (holgura positiva), o el porcentaje en que hay que disminuir los tiempos de ejecución para hacer que el sistema sea planificable (holgura negativa). Una holgura de cero significa que el sistema es justamente planificable y cualquier incremento de los tiempos de ejecución haría que dejara de serlo. Los diferentes tipos de holguras que se calculan son:

- *System slack*: afecta globalmente a todas las operaciones del sistema.
- *Processing resource slack*: afecta sólo a las operaciones ejecutadas en un recurso de procesamiento.
- *Transaction slack*: afecta sólo a las operaciones que se usan en una transacción.
- *Operation slack*: afecta sólo a una operación.

Las holguras se calculan modificando los tiempos de ejecución de peor caso de las operaciones y repitiendo el análisis hasta encontrar un punto en el que el sistema cambia su planificabilidad, esto se logra en pocas iteraciones mediante una búsqueda binaria. Esta técnica se apoya en la rapidez de las herramientas de análisis y supera las limitaciones de dependencias de flujo y falta de soporte para *jitter* que presentan otras tales como las propuestas de (Klein, *et al.,* 1993).

5. PERFILES Y HERRAMIENTAS PARA GENERAR EL MODELO

En el apartado 2 se ha planteado que a fin de conseguir en MAST una metodología de modelado abierta, fácilmente extensible y neutra respecto de la metodología de diseño, se ha optado por definir elementos de modelado que son muy próximos a los recursos de programación que proporciona la tecnología habitual de implementación de sistemas de tiempo real. Esta decisión conduce a que:

- Cada modelo describe únicamente una situación de tiempo real, y en consecuencia, un sistema que tenga diferentes estados de operación requerirá múltiples modelos.
- Utiliza como guía el modelo de transacciones y no hace referencia a la arquitectura lógica de la aplicación, lo cual es razonable porque corresponde al modelo que el diseñador tiene de su sistema de tiempo real, pero para construirlo debe diseccionar el modelo lógico basado en módulos software y localizar las operaciones que se emplean en el modelo de transacciones.
- El modelo está constituido por instancias, lo que hace que se repliquen los patrones de modelado al corresponder a instancias de una misma clase.

Como consecuencia de ello, el modelo de un sistema de tiempo real complejo está constituido por un gran número de elementos, lo cual presenta problemas en su gestión, y sobre todo hace difícil mantener la correspondencia entre los elementos del modelo de tiempo real y los componentes lógicos que constituyen la aplicación.

La solución a este problema se ha buscado a través de la definición de perfiles específicos para las diferentes metodologías y tecnologías de diseño. Un perfil define un nuevo conjunto de primitivas de modelado con un nivel de abstracción más alto, en las que se incorporan los patrones de modelado que son propios de la metodología que se utiliza. De esta forma el modelador maneja conceptos que son más próximos al modelo lógico que desarrolla y le permite mantener de forma más directa la correspondencia entre ambos.

Cada perfil que se define requiere un conjunto de herramientas, que “compilan” el modelo basado en el perfil al basado en el modelo MAST y recuperan asimismo los resultados de las herramientas sobre el modelo de alto nivel.

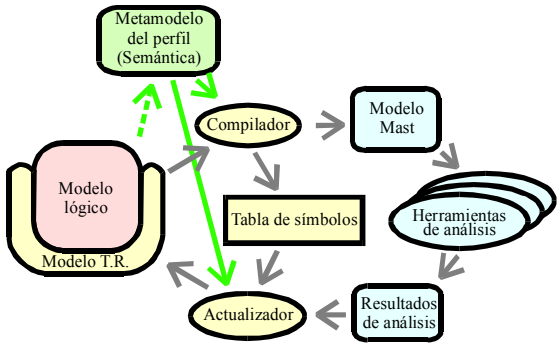

Figura 12. Herramientas asociadas a un perfil

La figura 12 muestra las herramientas que van asociadas a la definición de un perfil. Actualmente se han definido tres perfiles:

UML_Mast: Este perfil facilita el modelado de tiempo real de aplicaciones desarrolladas utilizando tecnologías orientadas a objetos y que se representan utilizando modelos UML (Medina, *et al*., 2001). Las características específicas que incorpora este perfil son:

- Describe de forma homogénea la vista lógica y el modelo de tiempo real haciendo uso de una herramienta UML estándar.
- Asocia los componentes de modelado de tiempo real y los resultados de los análisis a los módulos lógicos a que corresponden.
- Hace uso de descriptores que modelan elementos pertenecientes a clases lógicas y que posteriormente permiten generar instancias de modelado que describen el comportamiento de objetos que resultan de instanciar las clases.
- Define nuevos elementos de modelado con mayor nivel de abstracción.

ADA_Mast: Es un perfil que facilita el modelado de aplicaciones de tiempo real y distribuidas desarrolladas utilizando el lenguaje de programación ADA'95 y sus anexos D y E (Medina, *et al*., 2002). Este perfil aporta:

- La gestión implícita de la concurrencia a través de componentes tipo *Task*, así como de los mecanismos de sincronización asociados con ellos.
- Los recursos compartidos son implícitos a la declaración de los componentes del tipo *Protected* tal como ocurre en el lenguaje Ada.
- La comunicación entre componentes Ada distribuidos es implícita a la declaración de componentes del tipo *RCI*.

CBSE_Mast: Facilita el desarrollo del modelo de tiempo real de aplicaciones basadas en componentes (Drake, *et al*., 2002). Aspectos específicos que aporta el perfil son:

- Se describe el comportamiento y no necesariamente la estructura interna de los componentes software.
- Se basa en la generalización de los conceptos de "Descriptor" y de "Instancia", lo que le proporciona mejores características de componibilidad.
- Permite gestionar los elementos de modelado a partir de las bases de datos que contienen las descripciones de los componentes.

Las extensiones del entorno MAST a través de perfiles son mucho más fáciles de realizar que las basadas en la extensión directa del modelo MAST, pues no requieren la actualización de las herramientas al nuevo modelo. Sin embargo, es menos flexible ya que requiere que el modelo que introduce el perfil sea "compilable" al modelo base MAST.

6. CONCLUSIONES

El entorno MAST ha sido desarrollado con tres objetivos: proporcionar al diseñador industrial de sistemas de tiempo real un entorno en el que puede encontrar las principales herramientas de análisis y diseño de tiempo real con capacidad de manejar los sistemas de complejidad mediana y alta que actualmente tiene que desarrollar, ofrecer al investigador de tiempo real una plataforma abierta en la que con un esfuerzo reducido puede probar o comparar los métodos y herramientas que está desarrollando, y proporcionar al proveedor de componentes y herramientas de tiempo real un modelo homogéneo con el que especificar sus productos.

Las primitivas de modelado en que se basa esta metodología se corresponden con los recursos y mecanismos que ofrecen los lenguajes de programación, el hardware y los sistemas operativos, por lo que se da lugar a modelos relativamente amigables, fáciles de mantener a lo largo del ciclo de vida, así como a patrones de modelado reusables.

El entorno y los modelos son ofrecidos como abiertos a futuras extensiones por parte de los usuarios. Se basa en elementos de modelado que se definen a partir de un pequeño grupo de elemento raíces abstractos que gradualmente se van especializando en elementos concretos al incorporar las nuevas características que son propias de la plataforma, del sistema operativo, del lenguaje o de la tecnología específica de implementación que se utilicen.

Si comparamos MAST con otros entornos de desarrollo de sistemas de tiempo real, éste presenta un conjunto de características que lo hace muy competitivo, en particular para sistemas distribuidos. Estas características son:

- Define independientemente los modelos de la plataforma, de los componentes software y del modelo reactivo de la aplicación de tiempo real. Esta característica conduce a una alta reusabilidad de los modelos que se desarrollan.

- Utiliza un modelo no lineal de transacciones. Cada transacción puede ser activada por un conjunto de eventos externos y permite organizar las actividades de respuesta que generan mediante múltiples líneas de flujo de control en el que pueden existir alternativas, concurrencias, convergencias y sincronización entre ellas.
- Describe el comportamiento de los procesadores y de las redes de comunicación bajo el concepto abstracto común de recurso de procesamiento, lo que proporciona una gran uniformidad al tratamiento de los sistemas distribuidos.
- Admite una gran variedad de políticas de planificación en los procesadores y redes de comunicación (expulsables y no expulsables basadas en prioridades fijas, EDF basada en prioridades dinámicas, o servidor esporádico) y diferentes protocolos de sincronización en los recursos compartidos (techo inmediato de prioridad, herencia de prioridad, SRP).
- Utiliza transacciones generalizadas *end-to-end* que engloban secuencias de actividades que se ejecutan en diferentes procesos y procesadores.
- Define una rica variedad de tipos de requerimientos temporales: globales relativos a eventos externos o locales relativos al inicio de las actividades internas, relativas a plazos estrictos o laxos, relativos a *jitters* o a tasas porcentuales de cumplimiento de los plazos.

En su estado actual, el entorno está operativo para desarrollar aplicaciones que se realicen utilizando el lenguaje de programación Ada 95 con las extensiones de tiempo real y de sistemas distribuidos y/o basadas en sistemas operativos de tiempo real con API correspondiente al estándar POSIX. Asimismo, se han integrado herramientas que implementan los principales métodos de análisis y diseño de sistemas distribuidos de tiempo real que por su complejidad son difíciles de aplicar fuera de un entorno CASE. El entorno MAST es ofrecido con código abierto y es distribuido bajo licencia GPL de GNU. Puede descargarse de http://mast.unican.es/

Las principales líneas de desarrollo del entorno que actualmente se llevan a cabo son relativas a la extensión de la capacidad de modelado a sistemas basados en planificadores flexibles y jerarquizados, que hagan factible el desarrollo de aplicaciones distribuidas de tiempo real en entornos abiertos, en los que la aplicación que se desarrolla concurre con otras que no son conocidas, y para las que se puede requerir una cierta calidad de servicio.

Este trabajo ha sido financiado en parte por la Comisión Interministerial de Ciencia y Tecnología del Gobierno Español (CICYT) a través del proyecto de investigación con referencia TIC2002-04123-C03-02 (TRECOM), y por el programa IST de la Comisión Europea a través del proyecto de investigación FIRST con referencia IST-2001-34820.

REFERENCIAS

Audsley, N.C. (1991). *Optimal Priority Assignment and Feasibility of Static Priority Tasks with Arbitrary Start Times*, Dept. Computer Science, University of York, England..

Baker, T.P. (1991). Stack-Based Scheduling of Realtime Processes, En: *Journal of Real-Time Systems*, **Volume 3, Issue 1**, pp. 67-99.

Cheng, Albert M.K. (2002). *Real-Time Systems Scheduling, Analysis, and Verification*. ISBN 0-471-18406-3. John Wiley & Sons, Inc., New Jersey, USA.

Deng, Z. y J.W Liu (1997). Scheduling Real-Time Applications in an Open Environment. En: *Proceeding of the IEEE Real-Time System Symposium*. [9]

Douglass, B. P. (2000). *Real-Time UML: Developing Efficient Objects for Embedded Systems*. Addison-Wesley, USA.

Drake, J.M., J.L. Medina y M. González Harbour (2002). Entorno para el diseño de sistemas basados en componentes de tiempo real. En: *Actas del las X Jornadas de Concurrencia*. Jaca, España.

Gomaa, H. (2000). *Designing Concurrent, Distributed, and Real-Time Applications with UML*. Addison-Wesley, USA.

González Harbour, M., M.H. Klein, and J.P. Lehoczky (1994). Timing Analysis for Fixed-Priority Scheduling of Hard Real-Time Systems. En: *IEEE Trans. on Software Engineering*, **Vol. 20, No.1**.

González Harbour, M. and J.C. Palencia (2003). Response Time Analysis for Tasks Scheduled under EDF within Fixed Priorities, En *Proceedings of the 24th IEEE Real-Time Systems Symposium*, Cancún, México.

Gopalan, K. (2001). *Real-Time Support in General Purpose Operating Systems*, *ECSL Technical Report TR92*. Experimental Computer Systems Labs, Computer Science Dept, Stony Brook University, Stony Brook, NY - 11794-4400.

Gullekson, G. (2000). *Designing for Concurrency and Distribution with Rational Rose RealTime*. Rational Software White Paper.

Gutiérrez García, J.J. and M. González Harbour (1995). Optimized Priority Assignment for Tasks and Messages in Distributed Real-Time Systems. En: *Proceedings of the 3rd Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, California, pp. 124-132.

Harter, P.K. (1984). Response times in level-structured systems. En: *ACM Transactions on Computer Systems*, **vol. 5, no. 3**, pp. 232-248.

IEEE, The Institute of Electrical and Electronics Engineers. (2003). *IEEE Std 1003.13™-2003*. IEEE Standard for Information Technology—Standardized Application Environment Profile (AEP)—POSIX® Realtime and Embedded Application Support.

ISO, International Organization for Standardization (1995). *Ada Reference Manual ISO/IEC 8652*.

Joseph, M. and P. Pandya (1986). Finding Response Times in a Real-Time System. En: *The Computer Journal*, British Computing Society **29, 5**, pp. 390-395.

Kabous, Laila (2002). *An Object Oriented Design Methodology for Hard Real Time Systems: The OOHARTS Approach*. Tesis Doctoral, Escuela Carl von Ossietzky, Universität Oldenburg.

Klein, M., T. Ralya, B. Pollak, R. Obenza, and M. González Harbour (1993). *A Practitioner's Handbook for Real-Time Systems Analysis*. Kluwer Academic Pub., 1993

Kopetz, H., Zainlinger, R., Fohler, G., Kantz, H., Puschner, P. y Schutz, W. (1991). The design of real-time systems: from specification to implementation and verification. En: *Software Engineering Journal*, **vol. 6, no. 3**, pág. 72-82.

Lehoczky, J.P. (1990). Fixed Priority Scheduling of Periodic Tasks Sets with Arbitrary Deadlines. En: *IEEE Real-Time Systems Symposium*.

Leung, J. and J.W. Layland (1982). On Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks. *Performance Evaluation* **2**, 237-50.

Liu, C.L. and J.W. Layland (1973). Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. En: *Journal of the ACM*, **Vol. 20, No. 1**, pp. 46-61.

Liu, J.W.S. (2000). *Real-Time Systems*. Prentice Hall, USA.

Medina, J.L., M.Gonzalez Harbour y J.M. Drake (2001). MAST Real-Time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems. En: *Proc. of the 22nd IEEE Real-Time Systems Symposium*. London, England.

Medina, J.L., J.J.Gutierrez, M. González Harbour y J.M. Drake (2002). Modelling and Schedulability Analysis of Hard Real Time Distributed Systems Based on Ada Component. En: *Proceeding of the Int. Conf. On Reliable Software Technologies, Ada-Europe'2002*. Viena, Austria.

OMG (2003). *UML Profile for Schedulability, Performance and Time Specification, Version 1.0*. OMG document formal/03-09-01.

Palencia Gutiérrez, J.C., J.J. Gutiérrez García, y M. González Harbour (1997). On the schedulability analysis for distributed hard real-time systems. En: *Proc. of the 9th Euromicro Workshop on Real-Time Systems*. Toledo, España.

Palencia J.C. and M. González Harbour (1998). Schedulability Analysis for Tasks with Static and Dynamic Offsets. En: *Proceedings of the 18th. IEEE Real-Time Systems Symposium*, Madrid, Spain.

Palencia, J.C. and M. González Harbour (1999). Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems. En: *Proceedings of the 20th IEEE Real-Time Systems Symposium*.

Palencia, J.C. and M. González Harbour (2003). Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF. En: *Proc of the 15th Euromicro conference on real-time systems*, Porto, Portugal.

Rajkumar, R., L. Sha, and J.P. Lehoczky (1988). Real-Time Synchronization Protocols for Multiprocessors. En: *IEEE Real-Time Systems Symposium*.

Rajkumar, R. (1990). Real-Time Synchronization Protocols for Shared Memory Multiprocessors. En: *Proceedings of the 10th International Conference on Distributed Computing*.

Redell, O. (2003) *Response Time Analysis for Implementation of Distributed Control Systems*, Doctoral Thesis, TRITA-MMK 2003:17, ISSN 1400-1179, ISRN KTH/MMK/R--03/17--SE, Suecia.

Rodríguez Hernández, P. y J.J. García Reinoso (2003). *Nota sobre la Implementación del mecanismo de herencia*, VI Jornadas de Tiempo Real, Gijón, España.

Selic, B., G. Gullekson y P.T. Ward. (1994) *Real-time Object Oriented Modeling*. ISBN 0-471-59917-4, John Wiley & Sons, Inc., USA.

Sha, L., R. Rajkumar, and J.P. Lehoczky (1990). Priority Inheritance Protocols: An approach to Real-Time Synchronization. *IEEE Trans. on Computers*.

Spuri, M. (1996a). Analysis of Deadline Scheduled Real-Time Systems. *RR-2772*, INRIA, France.

Spuri, M. (1996b). Holistic Analysis of Deadline Scheduled Real-Time Distributed Systems. *RR-2873*, INRIA, France.

Tindell, K.W., A. Burns, and A.J. Wellings (1992). Allocating Real-Time Tasks. An NP-Hard Problem Made Easy. En: *Real-Time Systems Journal*, **Vol. 4, No. 2**,. pp. 145- 166.

Tindell, K. (1994a) *Adding Time-Offsets to Schedulability Analysis*, Technical Report YCS 221, Dept. of Computer Science, University of York, England.

Tindell, K. (1994b) An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks. En: *Journal of Real-Time Systems*, **Vol. 6, No. 2**.

Tindell, K., and J. Clark (1994). Holistic Schedulability Analysis for Distributed Hard Real-Time Systems. *Microprocessing & Microprogramming*, **Vol. 50, Nos.2-3**, pp. 117-134, April 1994.