
Universitat d'Alacant
Universidad de Alicante
jisbd
return
VIII Jornadas
de Ingeniería del Software
y Bases de Datos
12-14 Noviembre 2003. Alicante
03
EDITORES DE JISBD
Ernesto Pimentel
Nieves R. Brisaboa
Jaime Gómez

I.S.B.N.:84-688-3836-5
D.L.: MU-2267-2003

Imprime Compobell, S.L.

Printed in Spain

Entorno CASE para el desarrollo de sistemas de tiempo real[1]

Julio L. Medina, Patricia López Martínez, José M. Drake

Grupo de Computadores y Tiempo Real, Universidad de Cantabria.
Av. De Los Castros s/n 39005 – Santander, España
{medinajl, lopezpa, drakej}@unican.es
http://www.ctr.unican.es/

Resumen. Los sistemas de tiempo real que actualmente requiere la industria son sistemas muy complejos constituidos por plataformas distribuidas con decenas de aplicaciones, algunas de las cuales presentan requerimientos de tiempo real estricto o laxo y casi todas requieren niveles de calidad de servicio preestablecidos. En esta comunicación, se presenta el entorno MAST que ha sido diseñado a fin de dar soporte tanto a la complejidad algorítmica que se introduce con las nuevas herramientas para el análisis y diseño de estos tipos de sistemas, como a la gestión de la información que genera el proceso de desarrollo. Se describe la metodología de modelado que utiliza, basada en la descripción del sistema como conjuntos de transacciones concurrentes de flujo, que interaccionan entre sí al acceder a recursos activos y pasivos comunes. Se muestra la arquitectura del entorno, soportada en la especificación formal, estandarizada y extensible de las estructuras de datos que describen los modelos y la información que se genera. Estas estructuras constituyen la base de la interoperatividad entre las herramientas instaladas en el entorno. Se enumeran las herramientas que actualmente están disponibles y se justifica su adecuación para el desarrollo de aplicaciones de tiempo real distribuidas desarrolladas con Ada o haciendo uso de sistemas operativos de tiempo real con interfaz POSIX 1003.1b, y 1.c. Por último, se describen los perfiles UML_Mast, Ada_Mast y CBSE_Mast, que especializan la metodología para facilitar la generación de modelos cuando se usan técnicas orientadas a objeto, Ada95 con las extensiones de los anexos D y E, y tecnologías basadas en componentes, respectivamente.

1 Sistemas de Tiempo Real

Los sistemas de tiempo real empotrados que se necesitan actualmente en la industria automovilística, aeroespacial, de telecomunicación, de electrodomésticos, etc., son muy complejos. Utilizan plataformas computarizadas distribuidas constituidas por decenas de nudos procesadores, y deben alojar un gran número de aplicaciones independientes o acopladas, muchas de las cuales tienen requisitos de tiempo real, bien sean éstos estrictos o laxos y otras exigen niveles preestablecidos de calidad de servicio.

[1] Investigación financiada por el Programa Nacional de Tecnologías de la Información y de las Comunicaciones: Proyecto TRECOM (TIC2002-04123-C03-02).

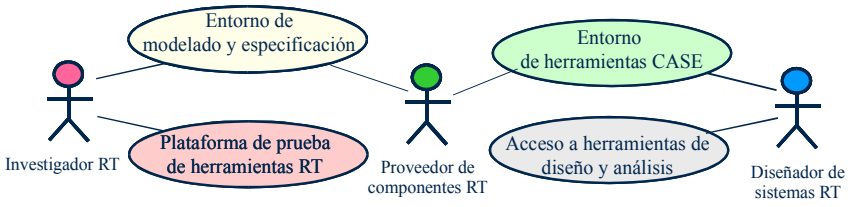

Fig. 1. Usuarios y casos de uso del entorno MAST.

Las herramientas que se emplean habitualmente para el análisis y diseño de aplicaciones de tiempo real son ineficientes para este tipo de sistemas, ya que sólo tienen capacidad de analizar la planificabilidad de la carga de cada procesador individualmente, y asignan prioridades tan solo a las tareas de tiempo real estricto, dedicando a las aplicaciones sin restricciones temporales los recursos residuales. Actualmente se dispone de herramientas y recursos que son especialmente adecuadas para el diseño de estos tipos de sistemas, que son de tiempo real, distribuidos y complejos:

- Actualmente se han propuesto [1][2] métodos de análisis de planificabilidad, de despliegue y de asignación de prioridades que permiten optimizar el diseño.
- Así mismo, se dispone [3][4] de sistemas operativos de tiempo real escalables y con interfaces estandarizadas que permiten estrategias de planificación flexibles y adaptadas que hacen posible redistribuir eficientemente los recursos para satisfacer tanto los requerimientos temporales estrictos y laxos, como los niveles de calidad.
- Y por último, se han propuesto [5][6] 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, etc., que hacen posible la planificación cuando la carga cambia en el tiempo.

Sin embargo, estas herramientas son de difícil aplicación tanto por lo complejo de sus algoritmos, como por la cantidad de información que debe ser procesada a lo largo de las sucesivas etapas del ciclo de desarrollo. Su uso requiere disponer de entornos CASE en los que se integren las diferentes herramientas y se facilite el intercambio de modelos y resultados entre ellas a través de bases de datos compartidas.

Nuestro grupo ha dedicado un gran esfuerzo durante los últimos años en el desarrollo del entorno MAST (Modeling and Analysis Suite for Real-Time Applications) [7][8]. Como se muestra en la figura 1, este entorno está destinado tanto a los grupos de investigación en métodos y herramientas de tiempo real como a los diseñadores de aplicaciones complejas de tiempo real en la industria, y también a los proveedores de recursos y componentes de tiempo real. El objetivo de esta comunicación es presentar la base conceptual y la estructura del entorno, las herramientas con las que está dotado y la forma de emplearlo en un ciclo estándar de desarrollo de sistemas de tiempo real.

2 Metodología de Modelado

El núcleo del entorno CASE es la metodología de modelado del comportamiento temporal de los sistemas, ya que constituye la base sobre la que operan las diferentes herramientas e intercambian información entre ellas. De entre los diferentes métodos

semi-formales de modelado de un sistema informático (hardware y software) propuestos, se ha adoptado la metodología basada en la descripción del sistema como conjuntos de transacciones concurrentes de flujo, que interaccionan entre sí por compartir recursos activos y pasivos que requieren ser usados en régimen de exclusión mutua. Esta elección se ha hecho por tres razones: es el método habitual que utilizan los diseñadores para la concepción del sistema de tiempo real, constituye el modelo respecto del que se han desarrollado los sistemas operativos y demás recursos de tiempo real que se utilizan, y además porque constituye el marco de referencia respecto del que se han formulado las principales herramientas de análisis y diseño para este dominio. La metodología coincide conceptualmente con la reciente propuesta del OMG [9] para estandarizar un marco de referencia para la interoperatividad entre herramientas en este campo, y de la que nuestro entorno puede considerarse una implementación tanto por sus objetivos como por la coincidencia en su base conceptual.

Cada instancia del modelo describe el comportamiento temporal de una "Situación de tiempo real", que representa un modo de operación del sistema con una determinada carga de trabajo y con un conjunto de requerimientos temporales establecidos. La "Situación de tiempo real" es el marco de especificación del sistema y el ámbito sobre el que operan las diferentes herramientas. Aunque en un mismo sistema se pueden presentar tanto múltiples situaciones de tiempo real, como otros muchos modos de operación sin requerimientos de tiempo real, el diseño de tiempo real del sistema consiste en garantizar que en cada uno de ellos, se satisfagan los requerimientos de tiempo real estrictos, laxos y las restricciones de calidad de servicio.

Por "Transacción" entendemos un conjunto de actividades relacionadas entre sí por dependencias de flujo que se desencadenan como consecuencia de un patrón de eventos externos o de temporización característico. Las transacciones tienen orígenes muy diversos:

- Modelan transacciones de tiempo real del modelo de negocio de la aplicación.
- Modelan procesos temporizados internos con requerimientos de tiempo real, introducidos en la fase de diseño de la aplicación para satisfacer algún requerimiento funcional o de calidad de servicio.
- Pueden representar artificios de modelado para representar requerimientos temporales entre eventos internos en secuencias cíclicas.
- Modelan operaciones concurrentes de background que aunque no tengan requisitos de tiempo real, sí interfieren con actividades de otras transacciones de tiempo real.

Aunque el concepto de transacción es habitual en el modelado de sistemas de tiempo real (Work threads, End-to-end jobs, Causal flow of activity, etc.), en el entorno MAST tiene las siguientes características no habituales, que le confieren una especial capacidad de modelado:

- Admite modelos de flujo muy generales, que incluyen operaciones de bifurcación (*fork*), sincronización (*join*), convergencia (*merge*) o distribución (*branch*).
- Puede ser de naturaleza distribuida, y englobar así líneas de dependencia de flujo que atraviesan múltiples nudos de procesamiento y redes de comunicación.
- Las transacciones son disparadas por conjunto de eventos lo que permite patrones de invocación y respuesta con muchas variaciones.

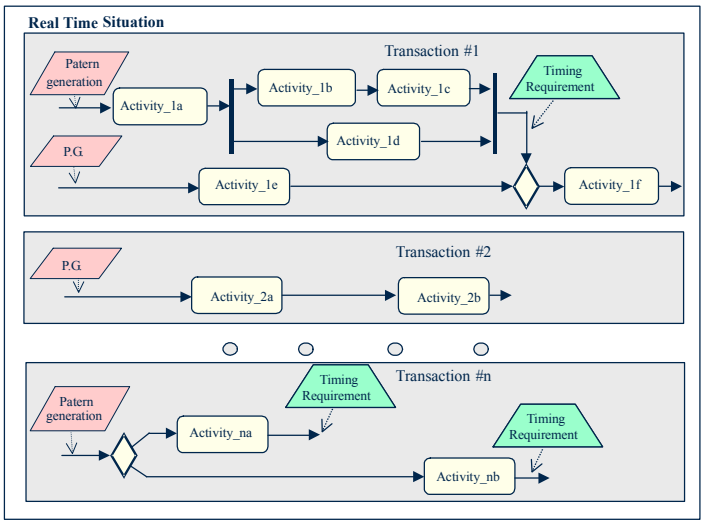

Fig. 2. Modelo de flujo de una Situación de Tiempo Real

En la figura 2, se muestra como una "Situación de tiempo real" se describe como el conjunto de transacciones que concurren en ese modo de operación. La carga de trabajo de la "Situación de tiempo real" se modela mediante los criterios de generación de eventos externos que se asocian a las transacciones. Cada "Generador de eventos externos" tiene asociado un patrón de generación característico (singular, periódico, esporádico, ráfaga, etc.) lo que permite modelar estadísticamente una rica y variada gama de situaciones de carga.

Así mismo, las transacciones sirven de marco de referencia para definir los "Requerimientos temporales" que se especifican en la aplicación. Estos requerimientos pueden clasificarse como globales si hacen referencia a la ejecución de una actividad respecto de un evento externo de disparo de la transacción, o como locales si hacen referencia al inicio de la propia actividad. Así mismo, los requerimientos pueden referirse a plazos estrictos, a los niveles de acierto de los plazos laxos, a los rangos de variabilidad ("jitter"), etc.

Cada "Actividad" representa la invocación de una operación que debe realizar el sistema como respuesta a los eventos (externos o internos) previos dentro de la transacción, y puede corresponder a acciones tan diferentes como la ejecución de un segmento de código en un procesador, la transferencia de un mensaje por un canal de comunicación, o la ejecución de alguna operación propia de un dispositivo hardware. La descripción independiente de la "Actividad", que representa la invocación dentro de una secuencia de flujo y la "Operación", que describe el modelo de temporización o de sincronización asociado con la ejecución de un código, independiza el modelo de un componente software de su uso dentro de diferentes transacciones.

Cada actividad de una situación de tiempo real está adscrita a un "Servidor de planificación" que modela el criterio de planificación con que se ejecuta. Las diferentes actividades asociadas a un mismo servidor de planificación son ejecutadas secuencialmente y son planificadas respecto a otras actividades adscritas a otros ser-

vidores de planificación de acuerdo con su naturaleza (expulsor, no expulsor, interrupción, servidor esporádico, etc.) y con los parámetros de planificación que éstos tienen asignados (prioridad). Cada servidor de planificación está asociado a un "Recurso de Procesamiento" que hace referencia al recurso activo (procesador, red de comunicación o dispositivo) en el que se planifican sus actividades.

Los "Recursos de procesamiento" modelan los recursos hardware/software que ejecutan las actividades que tienen asignadas y así mismo modelan las características asociadas a la gestión de su ejecución: capacidad de procesamiento del recurso, gasto de procesamiento por la gestión de la concurrencia (cambios de contexto) o de los relojes, la granularidad en la medida de tiempo, rangos de prioridades, etc. Los recursos de procesamiento se especializan en procesadores y redes de comunicación, en función de que tenga capacidad de ejecutar segmentos de código o de transferir mensajes. Aunque los parámetros asociados a procesadores y redes de comunicación tienen una semántica diferente (capacidad de procesamiento frente anchura de banda, cambio de contexto frente a protocolo de sincronización, proceso frente a sesión de comunicación, etc.) su tratamiento dentro de los modelos es semejante.

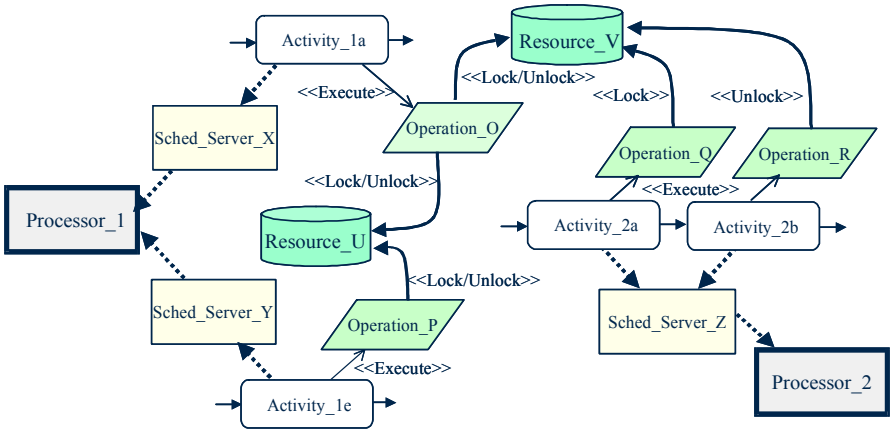

Fig. 3. Modelo de flujo de una Situación de Tiempo Real

Con esta metodología y con los recursos definidos se puede modelar una amplia gama de sistemas de tiempo real con la complejidad media y alta que se presenta en las actuales aplicaciones industriales. Los criterios que se han seguido para incorporar componentes de modelado en el entorno MAST han sido:

- Definir el conjunto de componentes como abierto. Para ello se ha definido una bien documentada y completa jerarquía de clases abstractas, que puede utilizarse para extender la capacidad de modelado hacia la especialización que se necesite.
- Sólo se incluyen las clases concretas que corresponden a situaciones y mecanismos que realmente existen y que se utilizan en los sistemas de tiempo real, por cuanto conducen a sistemas con comportamiento predecible y para los que existen métodos y herramientas de análisis y diseño.

3 Arquitectura del Entorno.

En la figura 4 se muestra los diferentes elementos que constituyen el entorno MAST. Se organiza alrededor de las estructuras de datos que representan el modelo y los resultados y trazas generadas por las herramientas. La definición formal, estandarizada, estable y documentada de las estructuras de datos es la base de la interoperatividad de las herramientas y de que pueda ser considerado el entorno como abierto a la incorporación de otras nuevas. Están formuladas en XML y formalizadas mediante las correspondientes definiciones "Schema". Esto representa una considerable ayuda a la validación, interpretación y transformación de la información ya que habilita para la extensión del entorno todos los recursos que proporciona la tecnología XML.

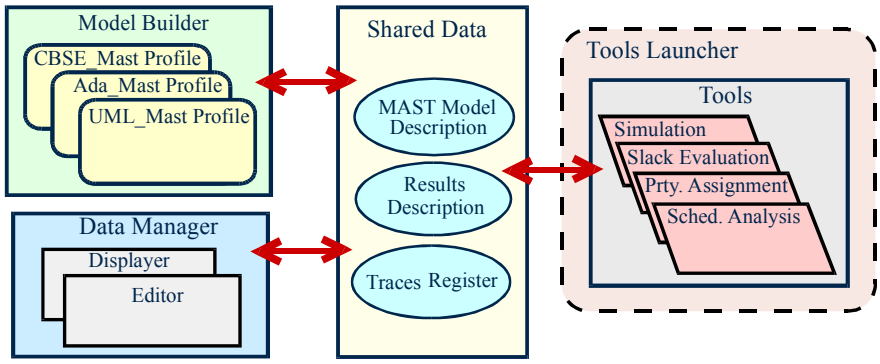

Fig. 4. Arquitectura del Entorno MAST

El entorno está dotado con 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:* Herramientas 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*: Herramientas 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.

4 Ciclo de Desarrollo y Herramientas de Análisis y Diseño.

El entorno MAST ha sido concebido como neutro respecto a la estrategia de diseño y al procedimiento de desarrollo que se utilice. La metodología de modelado propuesta se basa en identificar, de acuerdo con la carga de trabajo que debe ejecutar el sistema, los conjuntos de tareas que pueden concurrir en él, y relacionar con ellas las cargas de

procesamiento que requieren y los mecanismos de sincronización que se producen como consecuencia de que los recursos de procesamiento tienen capacidad limitada y de que los recursos pasivos deben ser accedidos en régimen de exclusión mutua. Como las primitivas de modelado con las que se describen estas características se basan en los mecanismos básicos que se incorporan en el código de la aplicación (software) y en los recursos que ofrece la plataforma (hardware y sistema operativo), se tiene capacidad de modelar cualquier estrategia de diseño formulada en función de ellos. Sin embargo, las herramientas con que debe estar dotado el entorno es función de las estrategias de diseño y del procedimiento de desarrollo que se utilicen.

La estrategia de diseño mas usada en la industria, en función de la que se ha desarrollado el conjunto de herramientas actualmente incluidas, es considerar que las aplicaciones tienen una naturaleza reactiva y están compuestas de conjuntos de procesos concurrentes que responden a los eventos generados por temporizadores o por interrupciones hardware externas. Esta estrategia es implementable utilizando sistemas operativos basados en prioridades estáticas y da lugar a sistemas cuya planificabilidad es analizable con métodos basados en la teoría de ritmo monotónico (RMA).

Tomando como referencia el proceso de desarrollo unificado [10], el ciclo de desarrollo de la aplicación está basado en cuatro fases cuyos objetivos son la identificación de las principales funcionalidades (*Inception*), la elaboración de las ideas básicas de la arquitectura (*Elaboration*), el desarrollo de la aplicación y la demostración de su capacidad funcional (*Construction*), y finalmente la verificación de que ésta satisface la especificación requerida y está dispuesta para ser utilizada (*Evaluation*). Cada fase se compone a su vez de una secuencia de iteraciones, cada una de ellas compuesta de cinco etapas: definición de requerimientos, análisis, diseño, implementación y prueba. El ciclo completo tiene una estructura en espiral, de forma que tras cada iteración se dispone de un prototipo analizable en el que se ha incrementado la funcionalidad, reduciendo así los riesgos que comprometen su operatividad.

Las herramientas disponibles en el entorno MAST para el análisis y diseño de tiempo real se pueden clasificar en cuatro grupos de acuerdo con sus objetivos:

- *Herramientas de análisis de planificabilidad*: que permiten establecer si las tareas que constituyen la aplicación pueden ser planificadas de forma que satisfagan las restricciones temporales establecidas. Estas herramientas son de naturaleza analítica y estudian en el peor caso el cumplimiento de los requerimientos temporales. Herramientas de este tipo actualmente disponibles son las clásicas de RMA[2] y las basadas en “offsets” [11] que son más eficientes y menos pesimistas.
- *Herramientas de análisis de holguras*: que proporcionan información de en cuanto podría incrementarse la carga de procesamiento de las tareas o reducirse la capacidad de procesamiento de los recursos manteniendo la aplicación aún planificable.
- *Herramientas de asignación óptima de prioridades*: que evalúan las prioridades optimas a asignar a los procesos, los techos de prioridad de los recursos compartidos y las prioridades que deben darse a los mensajes entre procesadores de forma que se obtengan las máximas holguras de planificabilidad. Actualmente están implementadas las denominadas linear HOPA [12] y templado simulado [13].

- *Herramientas de análisis de rendimiento*: que evalúan las características estrictas o laxas de respuesta temporal del sistema, los niveles de calidad de servicio y del uso

de los recursos en el sistema. A estos efectos, en la actualidad, se cuenta tan solo con herramientas de simulación, que requieren una cantidad de procesamiento bastante considerable.

Estas herramientas se utilizan en las sucesivas fases de diseño de la aplicación. En las fases iniciales, los modelos se basan en estimaciones heurísticas obtenidas de la experiencia y tienen por objeto traducir los requisitos temporales de las especificaciones en grados de concurrencia y presupuestos de tiempo formulados sobre la arquitectura del sistema y sirven de guía en las fases posteriores. En las fases de implementación se basan en estimaciones obtenidas del propio código que se genera y su objetivo es verificar la operatividad de la aplicación. Por último, en la fase de verificación, los modelos se basan en medidas obtenidas de las pruebas del sistema y su objetivo es validar y en su caso certificar el sistema.

5 Perfiles y herramientas para construcción del modelo.

La principal característica de la estrategia de modelado que se utiliza es que modela de forma independiente los diferentes elementos que influyen en el comportamiento temporal de una aplicación. Así por ejemplo, la capacidad de procesamiento que requiere un código se repercute sobre un componente de modelado tipo “Operation” que describe estadísticamente su magnitud, las características de sincronización con que se planifica se modelan mediante un componente de modelado tipo “Scheduling Server”, la granularidad temporal con que se activa, expulsa o finaliza se modela mediante un componente de modelado tipo “Timer”, los consumos de capacidad de procesado que requiere la gestión de su activación o posibles expulsiones son modelados por elementos de modelado del tipo “Processing Resource”, etc. Esto da lugar a modelos compuestos por múltiples patrones de modelado re-usables que se describen independientemente. El diseñador, puede construir el modelo con bloques previamente utilizados por otros diseñadores o por sí mismo en otras fases, y reducir su esfuerzo al ensamblar el modelo de forma incremental.

A este fin se han propuesto diferentes perfiles que definen nuevas primitivas de modelado en las que se incorporan los patrones de modelado que son propios de la metodología o de la tecnología de diseño que se usa, y se libera con ello al diseñador de tener que considerarlas. Cada perfil tiene una herramienta que procesa los modelos basados en él y genera de forma automática el modelo completo de la aplicación basado en las primitivas básicas. Actualmente se han definido tres perfiles:

UML_Mast: Facilita el modelado de tiempo real de aplicaciones desarrolladas utilizando tecnologías orientadas a objetos y que han sido desarrollas utilizando modelos UML [14]. Características específicas que incorpora este perfil son:

- Describe la vista lógica y el modelo de tiempo real sobre una misma herramienta CASE-UML desde la que se gobiernan todos los recursos del entorno MAST.
- Asocia los componentes de modelado de tiempo real y los resultados de los análisis a los componentes lógicos a que corresponden.
- Define nuevos elementos de modelado con mayor nivel de abstracción.

- Hace uso de descriptores que modelan elementos pertenecientes a clases lógicas, los cuales permiten posteriormente generar instancias de modelado que describen el comportamiento de objetos que resultan de instanciar las clases.

ADA_Mast: Este perfil aporta elementos de modelado de mayor nivel de abstracción, orientados a la representación de aplicaciones de tiempo real y distribuidas que se desarrollan utilizando el lenguaje de programación ADA'95 y sus apéndices D y E [15]. Sus características principales son:

- Facilita la gestión implícita de la concurrencia, a través de componentes tipo "Task", así como de los mecanismos de sincronización asociados con ellos.
- Los recursos pasivos son implícitos a la declaración de los componentes 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. Aspectos específicos que aporta el perfil son:

- Se describe el comportamiento y no sólo la estructura interna de los componentes.
- Se generalizan los conceptos de "Descriptor" y de "Instance".
- Permite gestionar los elementos de modelado a partir de las bases de datos que contienen las descripciones de los componentes.

6 Conclusiones y líneas de trabajo.

El entorno MAST que se presenta es una herramienta de trabajo con capacidad de ayudar al diseño de sistemas de complejidad mediana y alta, tal como se requiere en el desarrollo de las aplicaciones de tiempo real que actualmente emplea la industria. Se basa en una metodología de modelado de los sistemas de tiempo real que es muy familiar al diseñador por coincidir con la forma habitual con que él concibe su aplicación. Así mismo, los componentes de modelado reflejan exactamente los recursos y mecanismos que ofrecen los lenguajes con que se desarrolla el software, el hardware y los sistemas operativos de tiempo real, lo que da lugar a modelos fáciles de mantener a lo largo del ciclo de vida, así como a patrones de modelado re-usables.

La capacidad de modelado es ofrecida como abierta a través de una estructura de primitivas abstractas de modelado a partir de la que se pueden definir nuevos elementos especializados. Así mismo, está abierta a la definición de nuevos componentes de modelado de mayor nivel de abstracción que incorpore los patrones que son propios a una plataforma, a un sistema operativo o a una tecnología específica de diseño.

El entorno está aún en desarrollo, pero en su estado actual está operativo para desarrollar aplicaciones que se realicen utilizando el lenguaje de programación Ada con las extensiones de tiempo real y de sistemas distribuidos y/o basadas en sistemas operativos de tiempo real con API correspondiente a POSIX 1003.1b, y 1.c.

Actualmente se desarrollan varias líneas de investigación relativas a este entorno, tales como: extender la capacidad de modelado a sistemas basados en planificadores

flexibles que posibilitan el desarrollo de aplicaciones de tiempo real en entornos abiertos, desarrollar herramientas para el análisis de planificabilidad en sistemas basados en prioridades dinámicas, su aplicación como metodología de descripción de componentes de tiempo real, etc. Tenemos problemas para encontrar metodologías de análisis de sistemas que tienen requerimientos de tiempo real laxos o de calidad de servicio y que tengan capacidad de manejar aplicaciones de complejidad media y alta que habitualmente abordamos. Actualmente se utilizan herramientas basadas en simulación, pero la cantidad de procesamiento que requieren las hace poco eficientes.

El entorno MAST y la implementación actual del perfil UML-Mast son software libre y se ofrecen por tanto con código abierto, se distribuyen bajo licencia GPL de gnu y pueden obtenerse en http://mast.unican.es.

Referencias.

1. J.W.S. Liu: “Real-Time Systems”. Prentice Hall, 2000
2. M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, “A Practitioner's Handbook for Real-Time Systems Analysis”. Kluwer Academic Pub., 1993.
3. González Harbour and M. Aldea: “MaRTE OS: An Ada kernel for Real-Time Embedded Applications”. International Conference on Reliable Software Technologies, Ada-Europe’01. Leuven, May, 2001.
4. L. Abeni y G. Buttazzo: “ Integrating Multimedia Applications in Hard Real-Time Systems”. Proc. Of the 19th IEEE Real-Time Systems Symosium, 1998.
5. K. Gopalan: “Real-Time Support in General Purpose Operating Systems”
6. Z. Deng y J.W Liu: “Scheduling Real-Time Applications in an Open Environment” , En Proceeding of the IEEE Real-Time System Symposium, 1997.
7. J.M.Drake, M. Gonzalez Harbour, J.J.Gutierrez y J.C. Palencia: “Description of the MAST Model”. http://mast.unican.es/.
8. M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M. Drake: "MAST: Modeling and Analysis Suite for Real-Time Applications". Proceedings of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, IEEE Computer Society Press, pp. 125-134, June 2001.
9. B. Selic and A.Moore: “Response to the OMG RFP for Scheduling, Performance and Time: Revised Submission”. OMG document number: ad/2001-06-14.
10. I. Jacobson, G. Booch y J.Rumbaugh: “The Unified Software Development Process”. Addison Wesley, 1998.
11. J.C. Palencia y M. González Harbour: “Schedulability analysis for task with static an dynamic offsets” Proc. of the 19th IEEE Real-Time Systems Symposium, 1998.
12. K.Tindell y J.Clark: “Holistic schedulability analysis for distributed hard real-time systems”. Microprocessing &microprogramming, Vol.50, Nos. 2-3, pp. 117-134, 1994.
13. J.J. Gutierrez, J.C. Palencia y M. González Harbour:”Schedulability analysis of distributed hard real-time systems with multiple-event synchronization”, Euromicro Conference on Real-Time Systems, Stockholm, 2000.
14. J.L. Medina, M.Gonzalez Harbour y J.M. Drake: “MAST Real-Time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems”. 22nd IEEE Real-Time Systems Symposium (RTSS 2001). London, December 2001.
15. J.L. Medina, J.J.Gutierrez, M. González Harbour y J.M. Drake: "Modelling and Schedulability Analysis of Hard Real Time Distributed Systems Based on Ada Component" Proceeding of the Int. Conf. On Reliable Software Technologies, Ada-Europe’2002. Viena (Austria), June, 2002.