HERRAMIENTA PARA EL ANÁLISIS DE TIEMPO REAL ESTRICTO DE SISTEMAS AUTOMÁTICOS DE PRODUCCIÓN.

José M. Drake
drakej@ctr.unican.es

Julio L. Medina
medinajl@ctr.unican.es

Michael González Harbour
mgh@ctr.unican.es

Grupo de Computadores y Tiempo Real, Dpto. de Electrónica y Computadores.
Universidad de Cantabria

Resumen

Se presenta una metodología para modelar el comportamiento de tiempo real de sistemas automáticos de producción y un conjunto de herramientas que permiten analizar las prestaciones de tiempo real estricto que ofrecen los sistemas modelados. A través de estas herramientas se puede analizar si un sistema de producción bajo una situación de carga definida, cumple en el peor caso, un conjunto de requerimientos temporales estrictos que estén establecidos en su especificación. El uso de la herramienta es una solución fiable y eficiente para analizar sistemas complejos de tiempo real crítico, que de otra forma serían inabordables. La metodología y la herramienta que se presentan se fundamentan en el entorno "Mast" que está siendo diseñado por nuestro grupo dentro de un entorno CASE de desarrollo de sistemas software de tiempo real.

Palabras Clave: Tiempo real, sistema de producción automático, modelado orientado a objetos.

1 INTRODUCCIÓN.

Ciertos sistemas de producción críticos requieren cumplir de forma estricta un conjunto de requerimiento temporales para todas las condiciones de carga de su especificación. Estos sistemas son frecuentes en cadenas de producción robotizadas, en sistemas de control automático de vehículos y aviones, en sistemas de transportes, etc. En los sistemas realmente críticos, los análisis de tipo probabilísticos o los basados en simulación no son aceptables y se necesita utilizar métodos de análisis de peor caso, que certifiquen de forma absoluta que el sistema tiene capacidad de mantener su funcionalidad sin fallo, siempre que las entradas que alimentan al sistema se encuentren dentro de los márgenes especificados.

El análisis de peor caso de sistemas con la complejidad que suelen presentar las celdas industriales de producción, requiere un esfuerzo tan alto que en muchos casos es inabordable si no se dispone de las herramientas de análisis adecuadas. Las herramientas simplifican considerablemente el esfuerzo, ya que reducen el proceso de análisis a tan solo modelar el sistema con una metodología que sea compatible con la herramienta.

Actualmente nuestro grupo desarrolla un entorno de herramientas para el diseño de sistemas de tiempo real que hemos denominado "**Mast**" (Modeling and Analysis Suite for Real Time Applications). Este entorno está principalmente orientado al diseño de sistemas software de tiempo real, en los que tanto por la cantidad de componentes que suelen contener, como por la diversidad y sutilidad de los mecanismos de interacción entre ellos, conducen a complejidades muy altas. En este trabajo presentamos, una vista reducida del mismo, escalada de forma que, aunque tenga capacidad para abordar el análisis de sistema de producción complejos, sea lo suficientemente sencilla para manejarla sin esfuerzo.

Se propone una metodología de análisis de sistemas de tiempo real basada en tres fases:

1) Se modela el comportamiento de tiempo real del sistema que se analiza. El modelo se formula utilizando UML (Unified Modeling Language) y utilizando los componentes y las directrices del metamodelo que se ha defendido.

2) Empleando una herramienta que se ha desarrollado al efecto, se interpretan los símbolos gráficos y los datos del modelo UML establecido en la fase

anterior, obteniéndose la estructura de datos que requiere el entorno Mast.

3) Se analiza el sistema haciendo uso de las herramientas disponibles en el entorno Mast: Análisis de planificabilidad, análisis de holguras, asignación óptima de prioridades, cálculo de tiempos de contención en el acceso a recursos compartidos, etc.

Siguiendo este procedimiento, el ingeniero que realice el análisis de tiempo real de un sistema, deberá formular el modelo de tiempo real y ajustarlo cuantitativamente mediante medidas o estimaciones del sistema real, con la seguridad de que si ha seguido estrictamente el metamodelo, podrá obtener los resultados del análisis de forma automática utilizando la herramienta.

Este trabajo se orienta básicamente a presentar el metamodelo y a través de un ejemplo mostrar su aplicación. Información sobre el entorno Mast y sus herramientas se encuentra disponible en las referencias [1], [2], [3] y [4].

2 MODELO UML-MAST DE UN SISTEMA DE TIEMPO REAL.

La base de la herramienta que se presenta en este trabajo es el modelo que ofrece el entorno Mast. Este es un conjunto de componentes y reglas de conexión que permiten modelar con gran detalle y precisión el comportamiento dinámico y la temporización del sistema, reduciendo a la vez la complejidad del problema al ocultar los aspectos funcionales y operativos.

En este trabajo, el modelo de tiempo real se formula utilizando el lenguaje UML [5] y [6], que tiene una capacidad expresiva adecuada para este objetivo. Una ventaja que resulta de la utilización de UML, es que se disponen para él de diferentes herramientas de última generación que facilitan su uso inmediato y su adaptación si es necesario. La herramienta que habitualmente utilizamos y la que se emplea en el ejemplo que se presente en este trabajo es ROSE'2000e de Rational.

Un modelo de tiempo real **Mast-UML** es simplemente un conjunto de objetos y de relaciones entre ellos, que responde a una instanciación del metamodelo Mast-UML que se presenta. Este está compuesto por un número pequeño de clases de las que su semántica, sus atributos y las asociaciones entre ellas están definidas en él con todo detalle. Un modelo es un conjunto de objetos que son instancias de las clases del metamodelo, y que deben estar relacionados entre sí de acuerdo con lo establecido por las asociaciones definidas en él. Con su uso, se consigue por un lado representar de forma detallada el comportamiento dinámico de tiempo real del sistema, y por otro lado al seguir unas pautas tan formalizadas, se facilita su interpretación y análisis a través de herramientas automáticas.

El modelo de tiempo real que se propone, se compone de tres vistas complementarias, que en sus conjunto definen el comportamiento del sistema:

- Modelo de la plataforma: describe la capacidad operativa de los recursos hardware y software que constituyen el sistema.

- Modelo de los componentes funcionales: describe la temporización de las operaciones que se ejecutan en el sistema, así como los recursos, que por ser compartidos por varias operaciones en régimen de exclusión mutua, pueden inducir retrasos de espera en el acceso a ellas.

- Transacciones de tiempo real: describen las secuencias de eventos y operaciones que se pueden producir como respuesta a la ocurrencia de patrones de eventos externos, y que son la base de los requerimientos de tiempo real del sistema.

A fin de documentar de forma práctica la metodología de modelado que se propone, se presenta un ejemplo muy simple que ofrece los conceptos básicos.

Ejemplo: Sistema de conducción ferroviaria automática.

Se considera una red ferroviaria tal como la que se muestra en la figura 1. La red está divida en tramos (A, B, C, D, E y F), en los que por razones de seguridad (o de suministro de potencia) solo puede ser utilizado cada uno de ellos por un solo tren en cada momento. Así mismo, cada tramo se compone de un conjunto de sectores (a1, a2, a3, b1, ...) en cuyos límites existen sistemas de señalización y detección de paso. Las estaciones son puntos de la red en las que pueden coincidir varios trenes. Juegan un doble papel: son origen y destino de las rutas que siguen los trenes y así mismo, permiten que un tren que circula por un tramo, que sea a su vez requerido por otro tren más prioritario, pueda estacionarse en ella y cederle el paso.

La norma de conducción que se utiliza es muy simple: cada tren sigue de forma autónoma su ruta programada. Antes de acceder a un tramo, el tren debe esperar a que esté libre, y una vez que lo ocupa, imposibilita el acceso al mismo a cualquier otro tren. Si varios trenes están a la espera de acceder a un mismo tramo, accede aquel que tiene mayor prioridad. Cuando un tren llega a una estación

siempre libera el tramo. Si solo va de paso por ella, trata de acceder de nuevo al tramo, lo que conseguirá si no hay otro tren de mayor prioridad a la espera de acceder.

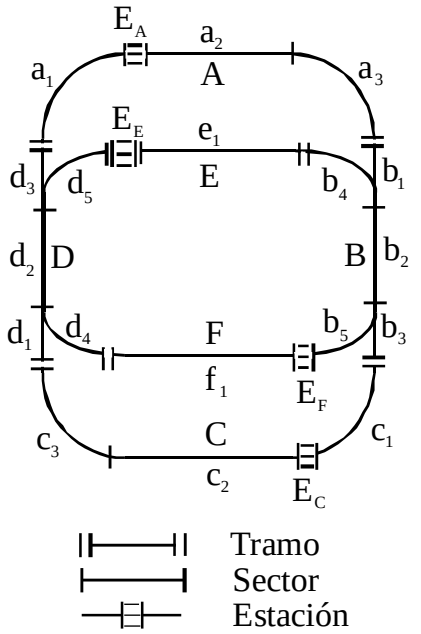

Figura 1: Red Ferroviaria.

En el ejemplo se analizará el siguiente caso de uso: Hay tres trenes en la red. El tren "Rojo" circula por el circuito exterior: sale cada 36 minutos de la estación Ea, va a la estación Ec y vuelve de nuevo a Ea. El tren "Verde" es un tren turístico que circula por el circuito interior: sale de la estación Ef cada 24 minutos, y vuelve a la misma estación sin necesidad de parar en Ee. Por último, el tren "Azul" recorre un circuito intermedio: sale de la estación Ec cada 36 minutos, va hasta la estación Ee, y vuelve de nuevo a la estación Ec. Las horas de partida de los tres trenes no están sincronizadas, y pueden ser cualesquiera.

Los requerimientos de tiempo real que se proponen, son que cada tren cumpla su horario, esto es, debe haber vuelto a su estación de salida antes de la hora de partida del siguiente recorrido que corresponde al siguiente periodo.

3 MODELO DE LA PLATAFORMA.

Incluye la declaración y la descripción de las capacidades operativas relevantes, a efecto de tiempo real, de los equipos y unidades funcionales que constituyen el sistema (Brazos, herramientas, sistemas de transporte, equipo de instrumentación, etc.), así como las sesiones operativas o procesos concurrentes dentro de las que se planifican las operaciones que se realizan.

En la figura 2 se muestra el metamodelo al que corresponden los modelos de plataforma de un sistema.

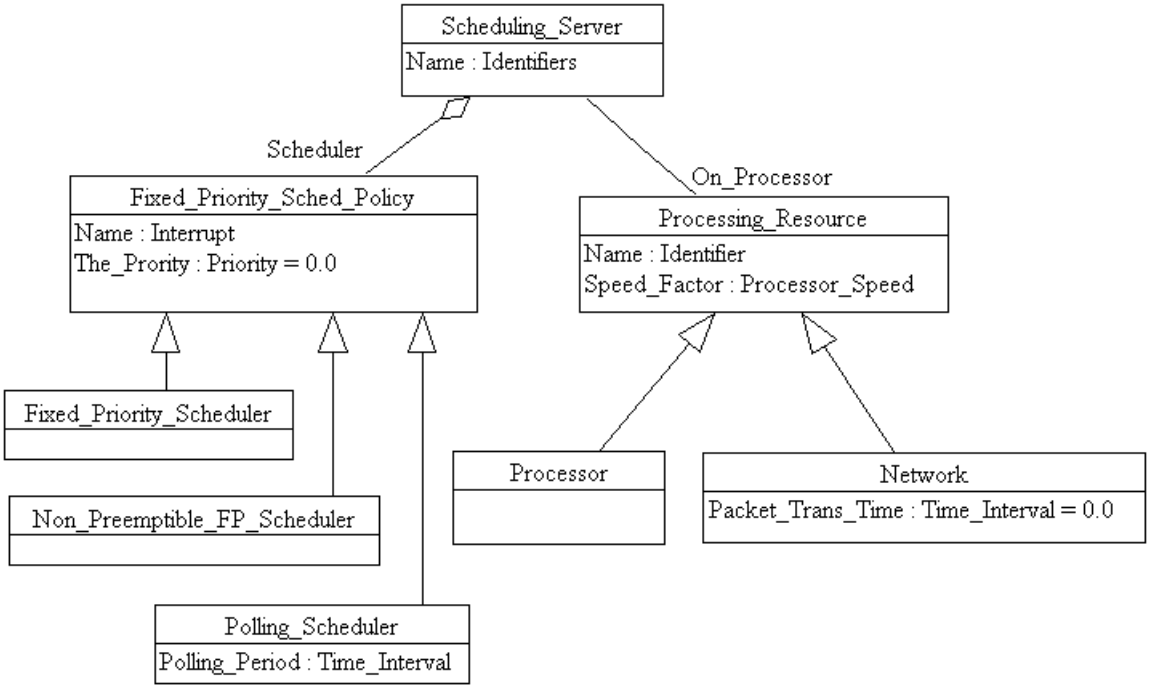

Figura 2: Metamodelo de la plataforma.

La clase central del modelo de la plataforma es la sesión operativa o proceso (**Scheduling_Server**) que representa las sesiones dentro de las que se ejecutan las actividades del sistema. Las diferentes actividades que se ejecutan dentro de una misma sesión se ejecutan secuencialmente, mientras que las operaciones de diferentes sesiones se ejecutan concurrentemente si están soportadas por equipos diferentes, o en concurrencia planificada si las sesiones están soportadas por un mismo equipo. Cada sesión tiene asociada una política de planificación (**Scheduler**), que describe el orden en que se ejecutan las actividades que se encuentren en espera de ejecución dentro de una misma sesión. La políticas de planificación para las que actualmente existen herramientas son las basadas en prioridades fijas (**Fixed_Priority_Sched_Policy**): Planificación no expulsora "**Fixed_Priority_Scheduler**", planificación expulsora "**Non_Preemtible_FP_Scheduler**", y planificación basada en escrutinio periódico "**Polling_Scheduler**".

Cada sesión tiene que estar asignada a un recurso de procesamiento (**Processing_Resource**), que representa el equipo o subsistema que ejecuta las operaciones. La velocidad operativa de un equipo está caracterizada por el atributo "**Speed_Factor**" que permite traducir las unidades normalizadas en que se expresa la duración de las operaciones a tiempo físico. Así mismo, de los recursos de procesamiento se han definido dos clases especializadas, los procesadores (**Processor**) que son los equipos que realizan operaciones productivas, y los canales de transporte (**Network**)que realizan operaciones de comunicación entre los equipos productivos.

El modelo de la plataforma se formula por uno o varios diagramas de objetos UML en los que cada objeto de tipo Scheduler_Server tiene asociado un objeto de tipo Scheduler_Policy que describe la política de planificación que sigue y un objeto de tipo Processing_Resource que establece el equipo que ejecuta las actividades.

Ejemplo: Red ferroviaria automática, modelo de la plataforma.

En el ejemplo previamente descrito, las actividades que se ejecutan consisten en recorrer sectores de acuerdo con las rutas establecidas, y esto es establecido por cada servicio: Estrella_Roja, Tren turístico, Ruta regional (que juegan el papel de proceso). Cada servicio es llevado a cabo por un tren concreto que ejecuta la actividad: Talgo_119, Eléctrico_1227 y Eléctrico_0883. Cada tren está caracterizado por su capacidad para ejecutar las actividades definidas. El que el factor de velocidad del Talgo_119 sea el doble que el del Electrico_1227 significa que es capaz de recorrer el mismo sector (ejecutar la misma operación) en la mitad de tiempo.

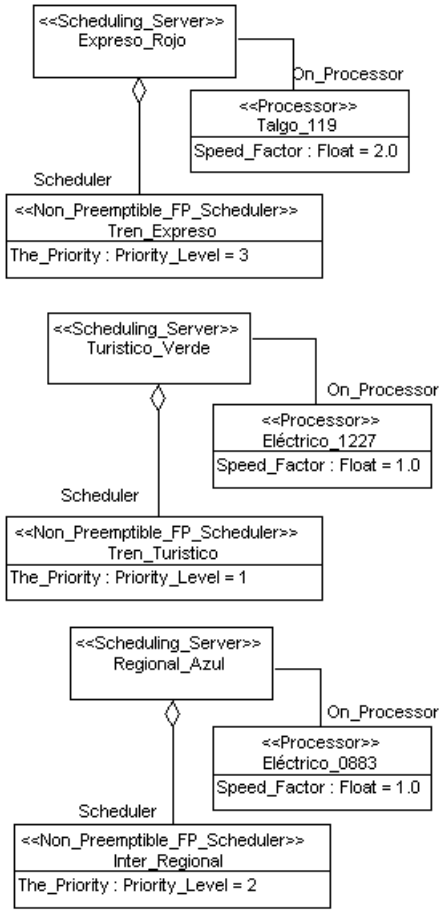

Figura 3: Modelo de plataforma de la red ferroviaria.

La política de planificación de los procesos, es de tipo no expulsora, lo que a este nivel no es relevante, ya que por la propia naturaleza del ejemplo en cada proceso (ruta que se sigue) solo hay una actividad activa (recorrer el siguiente sector) y no se requiere planificar su ejecución. La prioridad que se asigna sí es relevante, ya que aunque los trenes son independientes, deberán competir por recursos de acceso exclusivo (tramos de vía) y en caso de competir por ellos, se otorgará al de mayor prioridad.

4 MODELO DE LOS COMPONENTES FUNCIONALES

En esta segunda vista del modelo de tiempo real se caracterizan las operaciones que se pueden ejecutar en el sistema desde el punto de vista de su duración. Esto supone, describir la operación desde dos puntos de vista: en cuanto a la duración que requiere su ejecución material, y en cuanto al conjunto de

recursos que se requieren para poder ejecutarla. Como estos recursos pueden ser compartidos, y a ellos hay que acceder en régimen de exclusión mutua, la espera para su acceso será considerada por las herramientas de análisis para estimar el tiempo que transcurre entre la activación y la conclusión de la actividad.

En la figura 4 se muestra el metamodelo que define el modelo de tiempo real de los componentes funcionales.

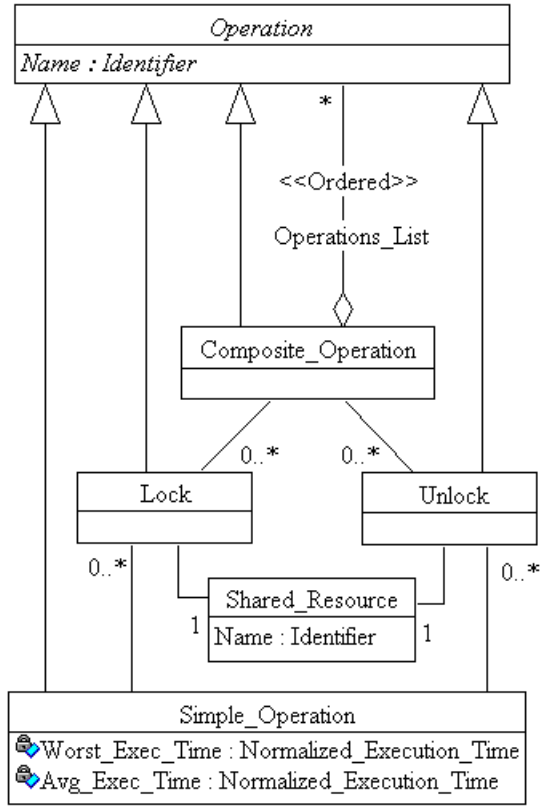

Figura 4: Metamodelo para la descripción de los componentes funcionales.

La clase básica de este modelo es operación (**Operation**) que representa una tarea que puede ser llevada a cabo por el sistema, una o varias veces, según proceda. Se definen dos tipos de operaciones especializadas. La operación simple (**Simple_ Operation**) que describe una actividad básica consistente en una reserva inicial de recursos, la ejecución de la tarea, y la posterior liberación de recursos y la operación compuesta (**Composite_ Operation**) que consiste en una lista de operaciones que se ejecutan secuencialmente, y que se introducen así sólo a efecto de simplificar la descripción de las operaciones complejas. La duración de cada operación simple, se expresa a través del atributo **Worst_Exec_Time**, para el peor caso, y **Avg_Exec_Time** para el caso promedio. Ambas se expresan en unidades normalizadas, lo que significa que la duración real de lo que tarda en ejecutarse una operación en un procesador concreto, se obtendrá dividiendo el valor normalizado por el factor de velocidad del procesador.

La segunda clase de este modelo es el recurso compartido (**Shared_Resource**) que representa un componente que debe ser accedido en régimen de exclusión mutua para satisfacer los requisitos funcionales de las operaciones que compiten por él. El acceso a los recursos compartidos es planificado, lo que significa que si varias operaciones se encuentran a la espera de acceder, se le concede al que sea de mayor prioridad. Para evitar la inversión de prioridad que pueden inducir los recursos compartidos, la ejecución de una operación mientras que tiene tomado el recurso puede cambiar su prioridad. En el caso de recursos planificados por techo de prioridad, la prioridad de la actividad que ha tomado el recurso es la mayor de entre la de los procesos que pueden acceder al recurso. En el caso de recursos planificados por herencia de prioridad, la prioridad de la actividad que ha tomado el recurso es la mayor de entre la de los procesos que se encuentran a la espera de acceder al recurso.

En UML el modelo de los componentes funcionales, se compone de dos tipos de diagramas. A través de diagramas de objetos se declaran los recursos compartidos y las operaciones definidas en el sistema. A su vez, cada operación simple o compuesta se describe mediante un diagrama de actividad asociado al objeto que lo declara, en el que se detalla la secuencia de operaciones simples, accesos a recursos y liberación de recursos que implementan su funcionalidad.

Ejemplo: Red ferroviaria automática, modelo de las operaciones

En este ejemplo, los tramos de vía constituyen los recursos compartidos. Para que un tren pueda recorrer un sector (realizar una operación) se requiere que haya accedido al tramo (recurso compartido), y esto solo podrá llevarse a cabo si ningún otro tren lo tiene tomado (está en él). Mientras que el tren se encuentra a la espera de poder acceder, el servicio (proceso) queda suspendido. En la figura 5, se muestra la declaración de los objetos que componen el modelo de las operaciones. Los recursos compartidos (Tramo_A, Tramo_B, ...) son objetos de la clase Shared_Resource.

Las operaciones corresponden a las tareas que pueden realizar los trenes en esta red. Se han definido como operaciones básicas, la tarea de recorrer los sectores (Recorre_A1, Recorre_A2, ...), y se han definido operaciones compuestas tales como

ir de una estación a otra. Por cada una de estas se realiza un diagrama de actividad, que describe la lista de operaciones simples, reservas de recursos y liberaciones de recursos que su ejecución conlleva.

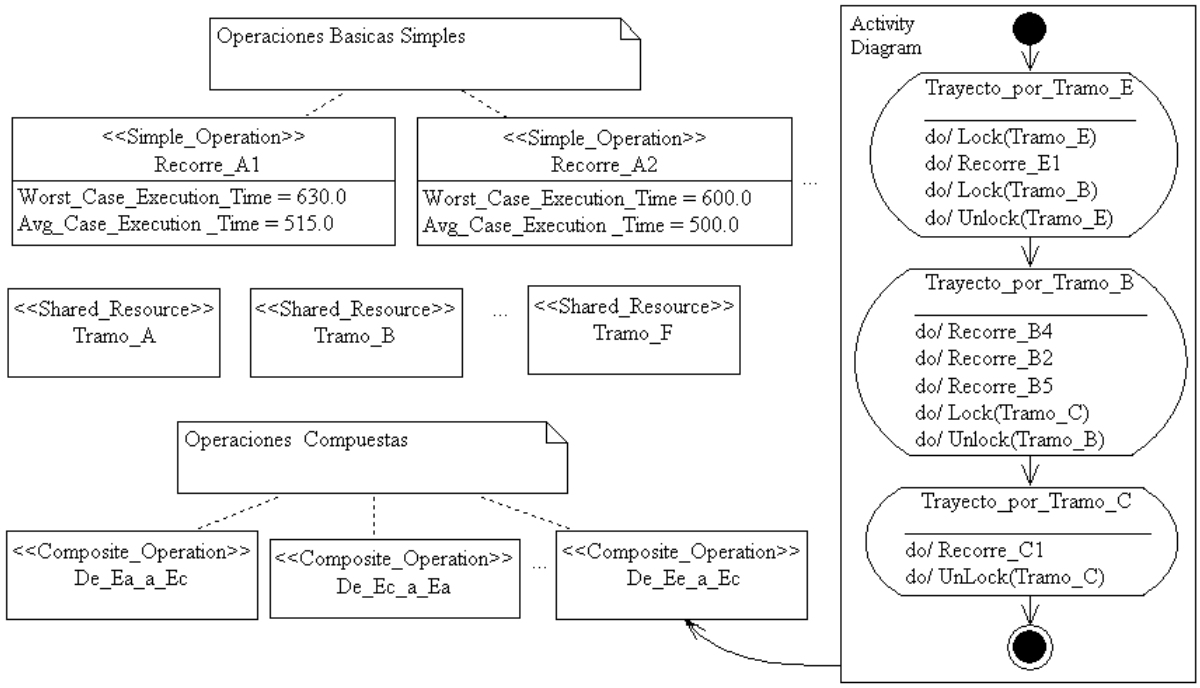

Figura 5: Modelo de operaciones de la red ferroviaria

5 TRANSACCIONES DE TIEMPO REAL

El objetivo de esta tercera vista del modelo es establecer los escenarios de eventos/actividades que describen la dinámica del sistema de tiempo real. Puesto que el objetivo del modelo de tiempo real es describir la temporización de la dinámica del sistema, no incluye aspectos funcionales, sino únicamente las secuencias de actividades que se desencadenan por cada patrón de eventos que representa el inicio de una transacción, así como la localización en ellas de

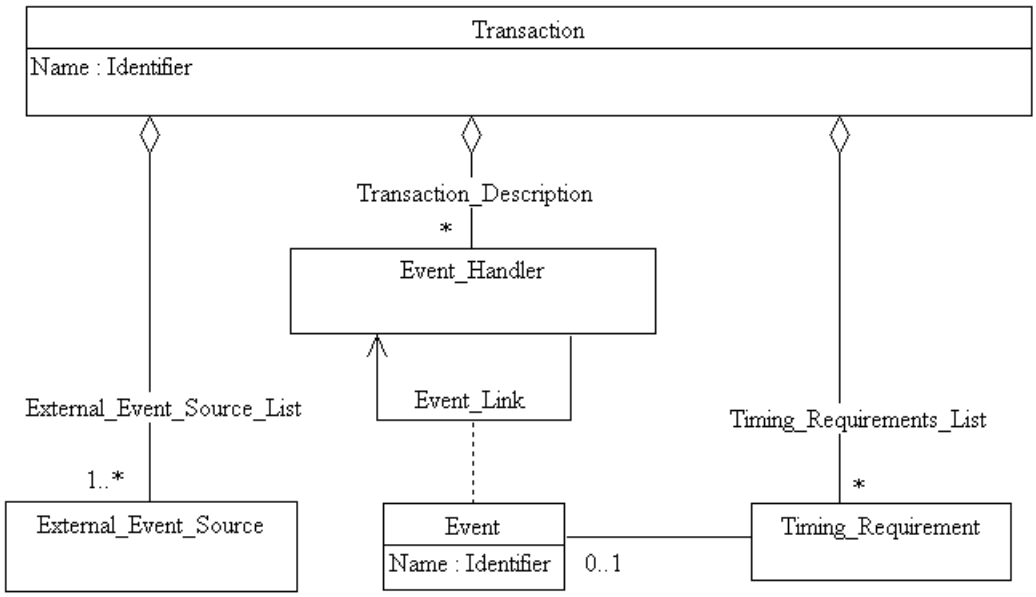

Figura 6: Metamodelo de las transacciones.

los eventos internos que representan estados de ejecución para los que se tengan establecidos requerimientos temporales. El modelo de transacciones de un sistema se describe mediante un conjunto de transacciones que coexisten en el sistema en un modo de funcionamiento dado. Este modelo se ajusta al metamodelo que se muestra en la figura 6.

Una transacción (**Transaction**) representa la secuencia de actividades/eventos que se desencadenan como consecuencia de un patrón de eventos externos de entrada que la dispara. Una transacción tiene asociadas tres listas de elementos, que en su conjunto constituyen la descripción de la transacción.

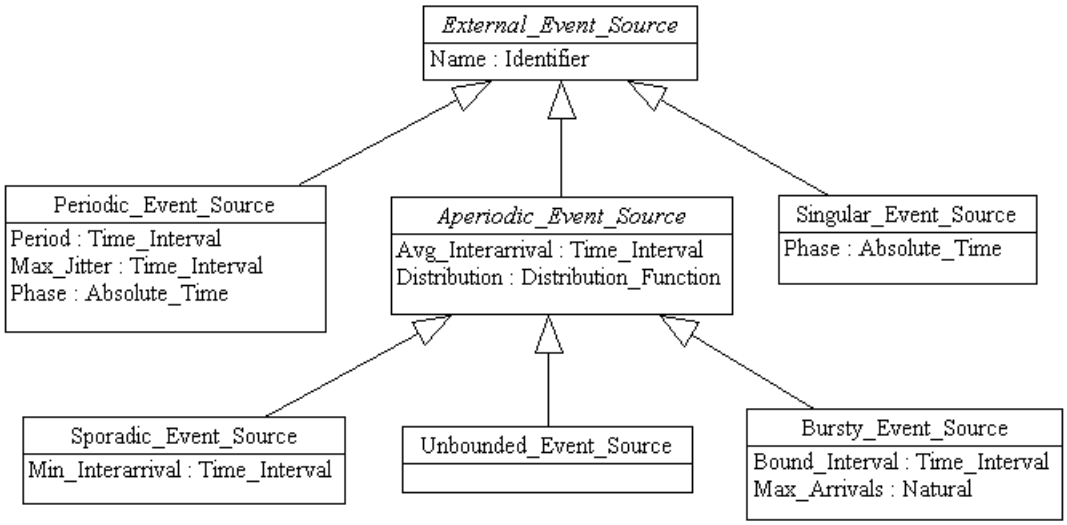

Figura 7: Metamodelo de las fuentes de eventos.

La lista de eventos externos (**External_Event_ Sources_List**) representa la declaración y la caracterización del conjuntos de eventos externos que disparan la transacción. Se han definido diferentes clases de fuentes de eventos externas (**External_Event_Source**) de acuerdo con el patrón de ocurrencia que presentan. En la figura 7 se muestra el metamodelo que define las fuentes de eventos externos definidos.

La lista de requerimientos temporales (**Timing_ Requirements_List**) representa la especificación de un requerimiento temporal que se requiere para un estado específico de la transacción. Por ello en el metamodelo el requerimiento temporal (**Timing_ Requirement**) se encuentra asociado a un evento interno (**Event**) que representa el estado de la transacción.

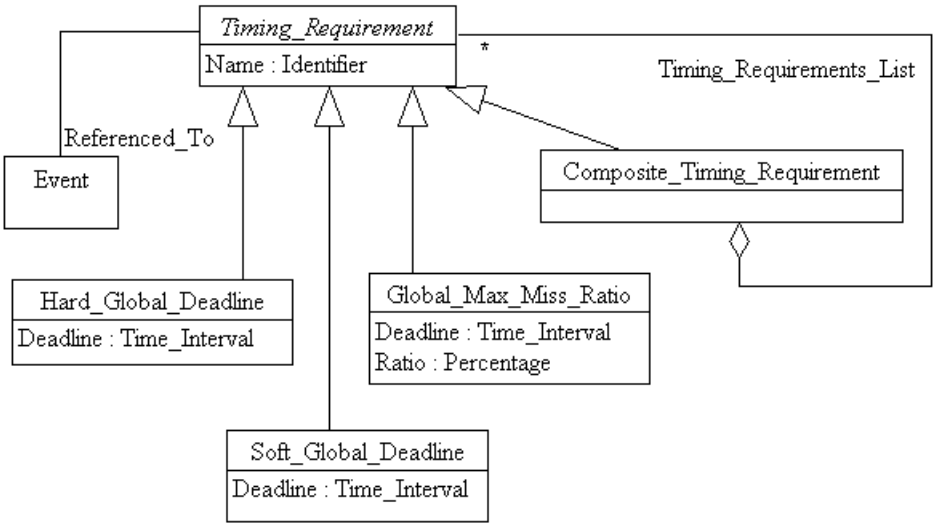

Figura 8: Metamodelo de los requerimientos temporales.

Se ha definido una amplio conjunto de clases de requerimientos temporales. En la figura 8 se muestra el grupo de aquellos que responden el estereotipo de requerimiento global. Estos son los que representan un requerimiento de plazo máximo para la ocurrencia del evento asociado, en relación al evento que se toma como referencia (**Referenced_To**).

La descripción de la propia transacción responde a una estructura de tipo grafo abierto, en el que los nudos son componentes dinámicos que responden a eventos de entrada y generan a su vez otros eventos de salida. A estos componentes los denominamos Manejadores de eventos (**Event_Handler**), los arcos representan a los eventos (estados instantáneos del sistema). La transacción representa básicamente secuencias de eventos (estados instantáneos relevantes del sistema), junto con las fuentes de eventos externos que inician las secuencias, las actividades que se ejecutan y otros conectores que establecen condiciones de activación de actividades o de generación de eventos.

En la figura 9, se muestra el metamodelo de los manejadores de eventos definidos.

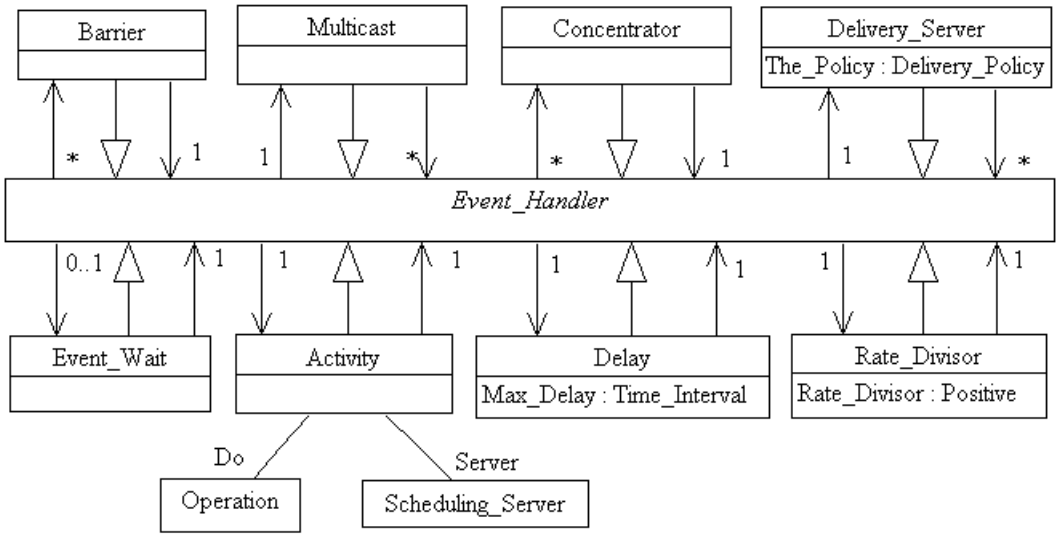

Figura 9: Metamodelo que define los tipos de manejadores de eventos.

La clase central de los manejadores de eventos es la actividad (**Activity**), que representa la ejecución de una operación dentro de la transacción. La clase **Wait_Event** representa la suspensión de una línea de flujo de la transacción a la espera de un evento externo. **Delay** representa la suspensión de una línea de flujo durante un intervalo de tiempo especificado. **Rate_Divisor** representa la necesidad de que ocurran varias eventos de entrada para que continúe el flujo. **Multicast** y **Barrier**, se utilizan para iniciar y concluir líneas de flujo concurrentes, y por último, **Delivery_Server** y **Concentrator** permiten establecer y concluir líneas de flujo alternativas.

En UML el modelo de transacciones se formula utilizando diagramas de objetos y diagramas de actividad. Los primeros se utilizan para declarar las propias transacciones y especificar los eventos externos que la provocan y los requerimientos temporales que se establecen en ella. Cada transacción debe ser descrita a su vez mediante un diagrama de actividad, en él los Event_Hanler se representan mediante actividades UML en las que su esterotipo indica la clase de Event_Hadler de que se trata, los eventos se representan mediante los enlaces UML, y los estados relevantes (con requerimiento temporal) se representan mediante estados instantaneos UML. La correspondencia entre fuente de eventos externos declarados en el diagrama de objetos y Wait_Event del diagrama de actividad se establecen porque ambos tienen el mismo identificador. De igual forma se establece la correspondencia entre requerimiento temporal y estado.

Ejemplo: Red ferroviaria automática, transacciones.

En este ejemplo, las transacciones representa las secuencias de actividades que se desencadenan despues de que se dá la salida a cada tren (Evento externo). Los únicos requerimientos temporales que se han definido es que cada tren este dispuesto para iniciar un nuevo recorrido cuando llegue la hora de la siguiente partida.

En la figura 10 se muestran algunos diagramas de descripción de las transacciones de este ejemplo. En el diagrama de objetos se declaran las tres transacciones (en este caso resultan independientes) Ruta_Roja, Ruta_Verde y Ruta_Azul. A ellas se asocian el correspondiente objeto que caracteriza el evento externo "Partida_xxx", y el correspondiente requerimiento temporal "Tren_xxx_Dispuesto".

Por cada transacción (en la figura 10 sólo se muestra la de la transacción Ruta_Roja) se debe incorporar el diagrama de actividad que la describe.

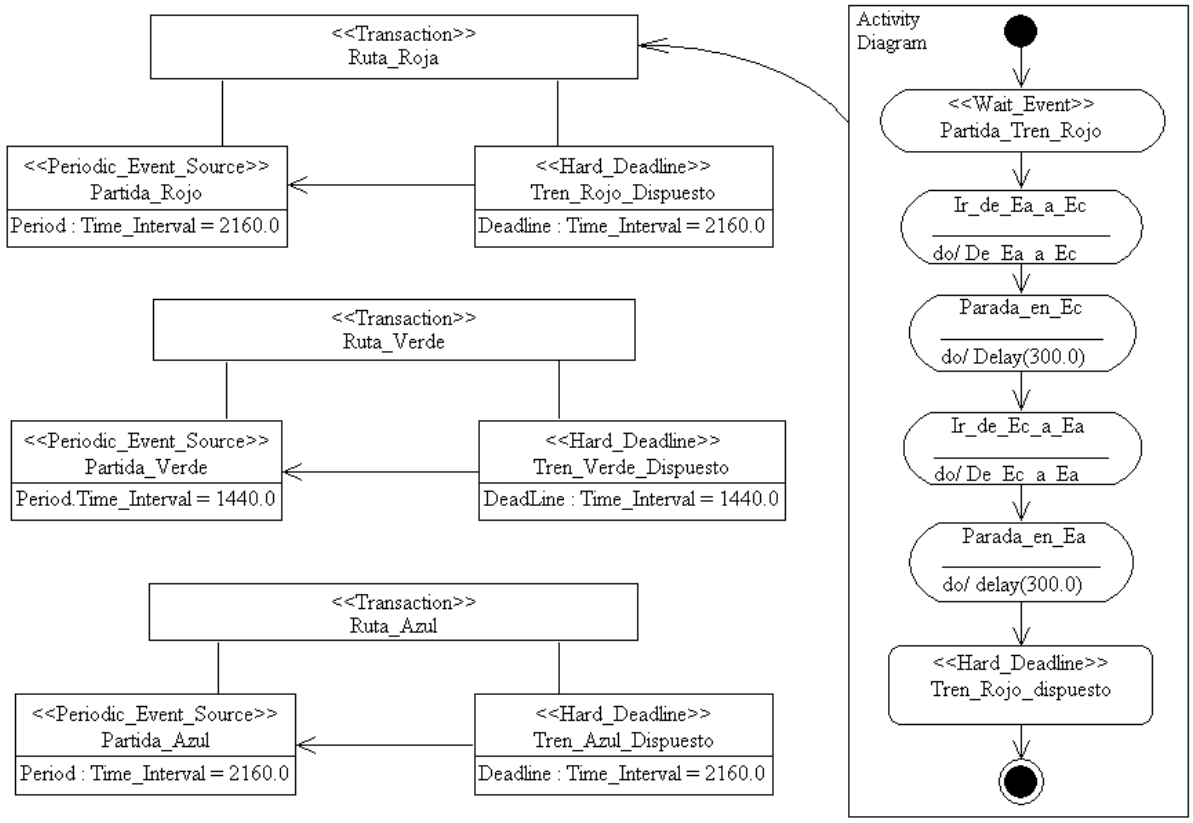

Figura 10: Transacciones de la red ferroviaria.

6 CONCLUSIONES

En este trabajo se ha presentado una metodología que permite modelar el comportamiento de tiempo real de sistemas dinámicos complejos. La metodología no está pensada para que pueda modelar todos los posibles sistemas que se puedan presentar (lo cual que nosotros sepamos no existe), sino que persigue representar de manera consistente aquellos para los cuales existan métodos de análisis. De hecho la herramienta esta concebida desde su inicio como extensible y hemos tratado siempre de extenderla tan sólo cuando se incorporan las correspondientes herramientas de análisis.

La metodología tiene una gran potencia de modelado, y como suele ocurrir en ingeniería, si se necesita disponer de un sistema crítico con requerimientos de tiempo real, es muy importante recordar cuando se diseña que al final debe ser certificado. Reconducir el diseño buscando que al final pueda ser analizable, es una estrategia muy razonable. En este sentido, la metodología de análisis que se propone puede ser entendida también como una pauta de diseño.

Nuestra previsión es poner el entorno Mast, a disposición pública en el mes de septiembre de 2000. Y dado que nuestro interés básico se centra en el diseño de software de tiempo real, dejamos abierta la utilización del mismo a grupos que quieran desarrollar su aplicación en otros campos. Para cualquier información o colaboración en este sentido póngase en contacto con los autores de este trabajo.

Agradecimientos

Este trabajo está financiado en el proyecto: "Diseño integrado de sistemas de tiempo real embarcados", Plan Nacional de Investigación(TIC99-1043-C03-03)

Referencias

[1] Gutierrez J.J. y González Harbour M.:"A Framework for Developing Distributed Hard Real-Time Applications" 25th IFAC Workshop on Real Time Programming, pp. 203-210, May, 2000.

[2] Gutierrez J.J., Palencia J.C. y González Harbour M.:"Schedulability Analysis of Distributed Hard Real-Time Systems with multiples Event Synchronization". 12th Euromicro Conference on Real-Time Systems. June, 2000

[3] Gutierrez J.J.:"Planificación, análisis y optimización de sistemas distribuidos de tiempo real estricto". Tesis Doctoral. Universidad de Cantabria, 1995

[4] Palencia J.C.: "Análisis de planificabilidad de sistemas distribuidos de tiempo real estricto basados en prioridades fijas".Tesis Doctoral. Universidad de Cantabria, 1999.

[5] Rumbaugh J., Jacobson I. y Booch G.: "The Unified Modeling Language: Reference Manual". Addison Wesley, 1999.

[6] Douglass B.P.: "Doing Hard Time: Developing Real-Time System with UML". Addison Wesley, 1999.