Metodología de modelado de sistemas de tiempo real orientada a la componibilidad[1]

Julio L. Medina, Patricia López Martínez, José M. Drake
Grupo de Computadores y Tiempo Real
Universidad de Cantabria
39006 Santander
medinajl@unican.es, lopezpa@unican.es, drakej@unican.es

Resumen

Se describe una estrategia para formular el modelo de tiempo real de una aplicación, que tiene las características de componibilidad necesarias para que el modelo de tiempo real de un sistema complejo resulte de la composición de los modelos de los componentes hardware y software con que se ha construido. Se basa en asociar a la descripción de los componentes un modelo parametrizado (RT-Descriptor_Model) que contiene toda la información cualitativa y cuantitativa sobre su comportamiento temporal, que se deriva de su código, y que define como parámetros, aquellas características que dependen del entorno de instanciación, así como las referencias a los modelos de tiempo real de otros componentes de los que requiere servicios. En el contexto de la ejecución de una aplicación, y dentro de cada modo de operación que se quiera analizar, se genera para cada instancia del componente que participa en la aplicación, un modelo de tiempo real completo (RT-Instance_Model) que una vez procesado por las herramientas de análisis proporciona información sobre los tiempos de respuesta de los servicios que ofrece la instancia.

1. Introducción

El modelo de tiempo real de un sistema informático es una abstracción que proporciona la información cualitativa y cuantitativa necesaria para evaluar y predecir su comportamiento temporal. Es el medio del que se vale el diseñador para formular los requisitos temporales durante la fase de especificación, razonar sobre su arquitectura en las fases de diseño y certificar su planificabilidad en las fases de validación [1][2].

Actualmente la industria utiliza plataformas distribuidas sobre las que se ejecutan concurrentemente múltiples aplicaciones, de las cuales algunas tienen requisitos de tiempo real y otras requieren determinados niveles de calidad de servicio. A fin de abordar la complejidad que resulta en estos sistemas, gestionar la diversidad de versiones que se han de manejar, y cumplir los plazos de desarrollo que impone la evolución del mercado, actualmente se ha impuesto la componentización de su diseño en todos sus niveles (figura 1): en los sistemas operativos, en el software de intermediación y en el diseño del código de la propia aplicación [3][4].

La componentización en sí es un aspecto complementario e independiente al proceso de diseño de tiempo real, sin embargo, al introducir cambios profundos en la metodología de desarrollo de las aplicaciones, interfiere con los métodos que han venido utilizándose en el diseño de tiempo real. En las metodologías tradicionales de diseño de sistemas de tiempo real [5][6][7], las aplicaciones son concebidas y descritas inicialmente como conjuntos de tareas o transacciones concurrentes dentro de un paradigma de sistema reactivo, y es en una fase posterior cuando, a fin de estructurar el código, las operaciones utilizadas se organizan en módulos siguiendo criterios de dominio, tareas o subsistemas. Por el contrario, cuando se utiliza una estrategia de diseño basada en componentes, el sistema se concibe y diseña ensamblando módulos reusables procedentes de catálogos de componentes disponibles, en función de que proporcionan la funcionalidad que se necesita, y es posteriormente cuando se asocian las secuencias de actividades resultantes a los procesos o hilos de flujo que introducen el modelo de concurrencia,

[1] Este trabajo ha sido financiado por la Comisión Interministerial de Ciencia y Tecnología del gobierno español dentro del proyecto de investigación TIC2002-04123-C03-02.

Actas del Simposio de Sistemas de Tiempo Real, pp.27-34
ISBN: 84-9732-448-X

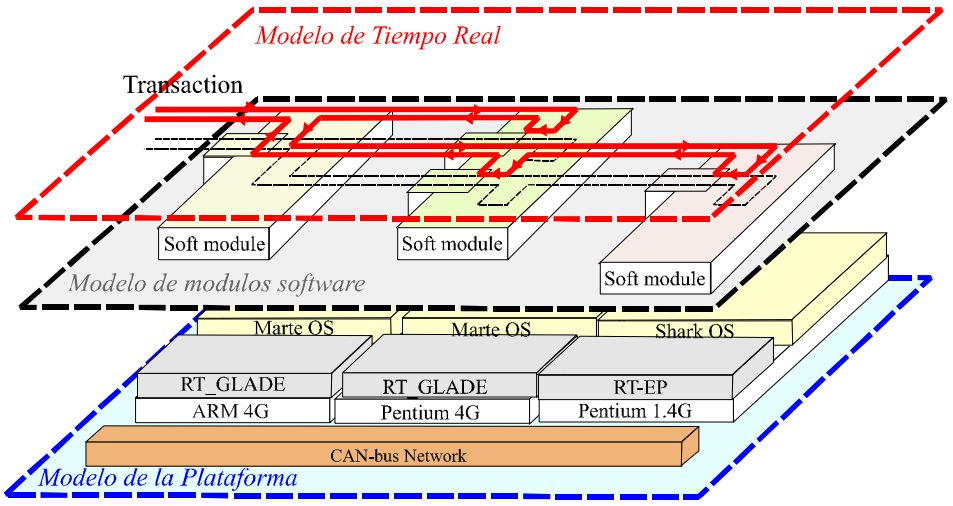

Figura 1: Niveles de modularización y modelado de aplicaciones de tiempo real

siendo no unívoca la asociación entre componentes y procesos. En el diseño de los sistemas basados en componentes hay que hacer compatibles los dos puntos de vista: el estructural (estático) que identifica las operaciones como servicios de las instancias de los componentes y el reactivo (dinámico) donde las actividades (invocación de las operaciones) se organizan por tareas o procesos.

Una metodología de modelado de aplicaciones basadas en componentes tiene que proporcionar recursos para formular el modelo de tiempo real de un componente como un ente independiente, y así mismo, debe ofrecer las propiedades de componibilidad necesarias para que el modelo de tiempo real de la aplicación pueda construirse por composición de los modelos de tiempo real de los componentes que la conforman.

Para ello hay que considerar las siguientes características que son inherentes a estos sistemas:

- El comportamiento temporal de un servicio, no solo depende del código del componente que lo ofrece, sino también del comportamiento temporal de los servicios de otros componentes de los que hace uso.
- La respuesta temporal de un componente software depende de la plataforma (hardware, sistema operativo, recursos de comunicación, etc.) que lo ejecuta, y por tanto, los modelos de tiempo real de los componentes han de incorporar referencias a los modelos de la plataforma.
- La respuesta temporal de un componente está condicionada por la ejecución de otros componentes que se ejecutan concurrentemente con él en la misma plataforma y con los que comparte recursos.

En trabajos previos [8] [9] se ha propuesto la metodología MAST (Modeling and Analysis Suite for Real-Time Applications) de modelado y análisis de sistemas de tiempo real distribuidos, basada en el modelado de cada situación de tiempo real como un conjunto de transacciones que se ejecutan concurrentemente compitiendo por recursos activos y pasivos compartidos.

En este trabajo se describe a nivel conceptual, una estrategia de formulación de los modelos de tiempo real con las características de componibilidad adecuadas para describir el comportamiento de sistemas de tiempo real basados en componentes. Aunque en este documento se presenta como una extensión de la metodología MAST, que hemos denominado CBSE-MAST, los conceptos y soluciones que se definen, transcienden de ella y pueden aplicarse a otras metodologías existentes[10][3].

2. Conceptos de descriptor e instancia de modelos de tiempo real

La extensión que se propone, dota a la metodología de la capacidad de modelar independientemente los subsistemas (hardware o software) que

denominamos componentes, a fin de que cuando se desarrollan sistemas que los utilizan, se pueda generar el modelo de tiempo real de los sistemas en función de los modelos de tiempo real de los componentes.

La clave de la propuesta está constituida por los conceptos de descriptor del modelo "*RT-Model_ Descriptor"*, e instancia de modelo "*RT-Model_ Instance"*.

Un *RT-Model_Descriptor* es un modelo parametrizado que contiene toda la información relativa al comportamiento temporal de un componente que es relevante para estimar el comportamiento temporal de cualquier instancia de ese componente con independencia de la aplicación en la que se utiliza, de la plataforma en que se ejecuta y del comportamiento de otros componentes que usa para implementar su funcionalidad.

Un *RT-Model_Instance* es un modelo completo susceptible de ser analizado, que describe el comportamiento temporal de una instancia del componente dentro del contexto de una situación de tiempo real o "*RT-Situation*" determinada.

Un *RT-Model_Instance* se genera a partir del *RT-Model_Descriptor* genérico del componente, asignando a sus parámetros valores y referencias a instancias concretas, que se deducen del contexto de la *RT-Situation* que se está modelando.

En la figura 2 se muestra un esquema para clarificar las diferencias entre estos conceptos. La sección superior representa elementos de información que pertenecen al catálogo de componentes. En ella, aparece registrado el componente software "Comp_A", y con él, junto a su descripción de uso (p.e. que ofrece la interfaz "Interfaz_ U") y a su descripción de instanciación (p.e. que requiere los servicios de un objeto que ofrezca la interfaz "Interfaz_R"), se encuentra también registrado su modelo de tiempo real "RT_Comp_ A_Model". En este contexto, el modelo es un *RT-Model_Descriptor* que contiene toda la información (derivada del código del componente) que es útil para estimar la repuesta temporal de cualquier operación en la que participe una instancia de él. Esta información no constituye un modelo completo analizable, ya que en ella se describe, por ejemplo, la cantidad de procesamiento que requiere la ejecución del código del procedimiento Proc_F(), pero de ella no se puede deducir el tiempo de ejecución de éste, ya que no se conoce la capacidad de procesamiento del procesador que ejecuta el código, ni tampoco el tiempo de respuesta de los procedimientos de la interfaz Interfaz_R, que se invocan en ella. El *RT-Model_Instance* si describe la información que falta para estar completo.

En la sección inferior de la figura 2, se representa el contexto que corresponde a la ejecución de una aplicación denominada "App_1", que hace uso de la instancia "obj_A" del componente "Comp_A". En ella se especifica que la instancia "obj_A" se ejecuta en el procesador "Proc_Q", y que es el objeto "obj_S" del tipo "Comp_RX" el que proporciona la funcionalidad de la interfaz "I_R" requerida. En este contexto, se puede generar su modelo "RT_obj_A_Model" del que las herramientas de análisis de tiempo real pueden estimar el tiempo de ejecución del procedimiento "obj_A.Proc_F()". "RT-obj_A_Model" es un *RT-Model_Instance* que se genera asignando valores concretos a las referencias no resueltas del descriptor "RT-Comp_A_Model", en este caso "RT-Q_Model" es el modelo del procesador que ejecuta la instancia y "RT_obj_S_Model" es el modelo de la instancia "obj_S" que utiliza.

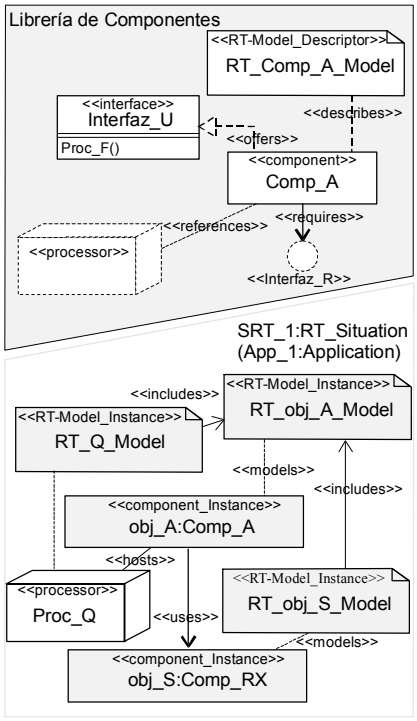

Figura 2: RT_Model_Descriptor y RT-Model_Instance

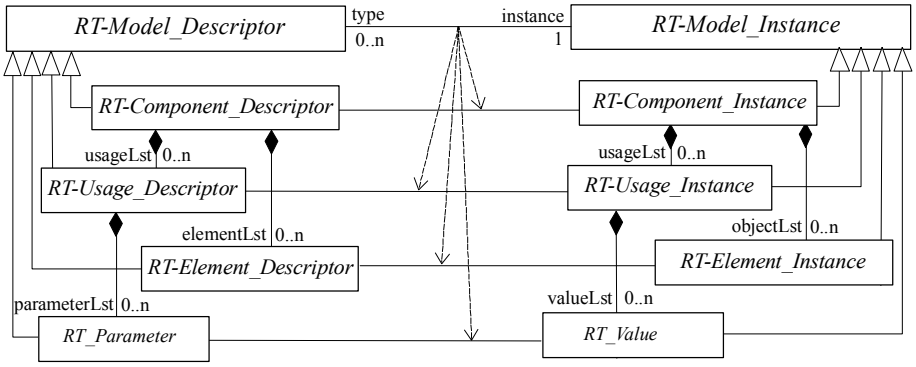

Figura 3.- Núcleo de la extensión de la metodología de modelado de tiempo real.

En la figura 3 se muestran las clases raíces del metamodelo que describe los modelos que se proponen.

Dentro de este metamodelo, un *RT-Model_Descriptor* es un ente abstracto de modelado que representa un descriptor genérico, y habitualmente parametrizable, que define la información que se necesita para describir el modelo de tiempo real de algún tipo de recurso o servicio del sistema. El descriptor proporciona la información semántica y cuantitativa que es común a todos los entes que puedan resultar de su instanciación. Un *RT-Model_Instance* representa el modelo de tiempo real concreto y final de una instancia individual de un recurso o servicio del sistema dentro de una situación de tiempo real. En el modelo de un sistema, cada objeto *RT-Model_Instance* se declara con referencia a un *RT-Model_Descriptor* que describe su estructura y semántica, y del que se deriva al asignar valores o referencias a instancias concretas a todos los parámetros que el descriptor tiene declarados.

Un *RT-Component_Descriptor* es un descriptor que modela un componente hardware o software. Desempeña una triple función:

- Constituye un elemento contenedor que declara:
  - Los atributos, parámetros o símbolos que deben ser establecidos para declarar una instancia del modelo. De cada uno de ellos se especifica su tipo y opcionalmente un valor por defecto.
  - Los componentes externos a él que deben ser referenciados (*Referenced*) ya que el modelo del componente hace uso de los modelos que representan.
  - Los componentes que forman parte de él (*Aggregated*) y que serán instanciados, por cada instancia suya que se declare en el modelo.
- Declara los modelos de las formas de uso del componente que corresponden a los servicios que ofrece el componente y que tienen prestaciones de tiempo real.
- Define un ámbito de visibilidad, para nombres de símbolos, atributos y componentes.

Una *RT-Component_Instance* es el modelo de un objeto hardware o software definido en el contexto de una *RT_Situation* que contiene la información cuantitativa completa, final y analizable de su comportamiento de tiempo real. Contiene:

- La asignación a cada parámetro de un valor concreto que esté definido dentro del modelo.
- La asignación de referencias a *RT-Component_Instance* concretas declaradas en el modelo.
- La declaración recursiva de todas las instancias de componentes que estaban declaradas como *Aggregated* en el *RT-Component_Descriptor* del que se deriva.

Un *RT-Element_Descriptor* es el modelo de un subcomponente que se declara en un *RT-Component_Descriptor* como parte del modelo de componente que describe. Existen tres tipos de declaraciones de elementos:

- *«aggregated»*, que declara el modelo de un componente que es una parte del modelo en el que se declara. Esto significa que por cada instancia del componente que lo declara, será instanciado un modelo del elemento agregado.

- *«referenced»*, que declara el modelo de un componente externo al que se describe y que necesita conocerse pues es parte del mismo. Su instanciación sin embargo no es implicada por la instanciación del componente que lo referencia.
- «declared» que define el modelo de un componente que se necesita conocer para construir el modelo del componente que se describe.

Un *RT-Element_Instance* es cada una de las instacias que tienen que definirse en el contexto de la *RT-Situation* para que el modelo de la instancia del componente esté completo. En la declaración de una *RT-Element_Instance* se deben asignar valores concretos a todos los atributos existentes en el *RT-Element_Descriptor* que no tengan definido un valor por defecto, y se podrá asignar también un nuevo valor a los atributos que tengan valor por defecto.

Un *RT-Usage_Descriptor* modela las características temporales de una forma de uso ("*Usage*") de un componente. Corresponde, por ejemplo, con la caracterización de la capacidad de procesamiento que requiere la ejecución de un procedimiento, o con la caracterización del consumo de la capacidad en un procesador que se realiza en respuesta a una interrupción.

Una *RT-Usage_Instance* modela el uso de los recursos que se realiza como consecuencia de la ejecución de un servicio concreto del componente en el contexto de una *RT-Situation*.

En este apartado se han descrito las clases abstractas de nivel más alto que constituyen la extensión de la metodología de modelado de tiempo real que se presenta en este trabajo. De ellas se derivan las primitivas concretas de modelado que describen el comportamiento temporal de los elementos que participan en la ejecución de la aplicación. En las referencias [8] y [9] puede encontrarse una descripción detallada de los aspectos que modelan y de los atributos que definen su comportamiento. Las primitivas de modelado se clasifican en tres tipos:

- Recursos: Modelan el comportamiento de los elementos de una aplicación que contribuyen a la capacidad de procesamiento disponible para ejecutarla. Unos porque proporcionan la capacidad (procesadores y redes de comunicación), otros porque la menguan (temporizadores, drivers y planificadores) y otros porque limitan el uso de la capacidad estableciendo exclusión mutua o sincronización (servidores de planificación y recursos compartidos).
- Formas de uso: Proporcionan la información que es útil para evaluar la temporización de la ejecución de las actividades que deben ejecutarse en la aplicación. Modelan la cantidad de procesamiento y los recursos que requiere para su ejecución. Tipos de forma de uso definidos son las transacciones que modelan conjuntos de actividades relacionadas por flujo de control dentro de un modelo reactivo de la aplicación y las operaciones que modelan la cantidad de procesamiento y los recursos que requiere una actividad.
- Situaciones de tiempo real: Modelan el comportamiento temporal de una aplicación en un determinado modo de operación. Son los elementos raíz de cualquier modelo completo analizable. Describen las instancias de los modelos de los recursos que participan en el modo de operación, de las actividades que se invocan, y su organización como un conjunto de transacciones concurrentes.

3. Formulación de los modelos y herramientas de composición

En la figura 4, se muestra la estructura de los elementos de más alto nivel que constituyen el modelo de una situación de tiempo real de una aplicación. La información se encuentra almacenada en dos tipos de ficheros, que respectivamente representan los descriptores asociados a los componentes y el modelo de la situación de tiempo real. Ambos se codifican utilizando información tipo texto estructurado en XML.

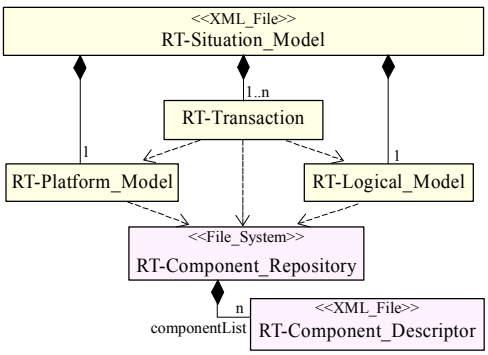

Figura 4.- Organización del modelo de una aplicación.

En la base de datos o *RT-Component_ Repository*, que contiene la información asociada a los componentes (funcional, de instanciación, de empaquetamientos, etc.) se encuentran también registrados sus modelos de tiempo real, que se almacenan como ficheros de tipo texto. Cada fichero codifica un *RT-Component-Descriptor*, que contiene todos los *RT-Element_Descriptor* y *RT-Usage_Descriptor* relativos a la descripción del comportamiento temporal del componente. Como ejemplo, en la tabla 1, se muestra una sección del modelo de tiempo real del componente software ADQ_9111 que es el software de control de una tarjeta comercial de entrada/salida de señales analógicas y digitales.

El modelo no sólo describe la cantidad de procesamiento que se requiere para realizar las operaciones que ofrece, como "DigRead", "DigWrite" y "SetBlinking", sino también el modelo de comportamiento de otros recursos que son introducidos por cada instancia del componente y que afectan a la planificabilidad de las aplicaciones que lo utilizan. Por ejemplo, "DigMutex" que modela el *mutex* que arbitra los accesos concurrentes a otras líneas digitales de la misma tarjeta, "BlinkingControl" que modela la tarea periódica que controla el parpadeo de las líneas o "BlinkingServer" que modela el servidor de planificación de esa tarea. La cantidad de procesamiento que requiere cada operación, se describe mediante magnitudes de tiempo de ejecución normalizadas, que podrán traducirse a tiempo físico cuando se conozca el procesador que lo ejecuta (el tiempo físico de ejecución se obtiene dividiendo los tiempos normalizados de ejecución por el valor del *SpeedFactor* del procesador). El modelo de este componente tiene definido tres parámetros a los que se han de asignar valores deducidos del contexto de ejecución, cuando se genere el modelo de una instancia del componente: "@Host" que es la referencia al modelo del planificador (procesador) en el que se ejecuta, "@PollPrty" que es la prioridad del *thread* que controla autónomamente el parpadeo y "@BlinkingTime" que corresponde al periodo de parpadeo. Este último parámetro tiene asignado el valor por defecto 1.0, por lo que no es necesario que se le asigne un nuevo valor si el periodo de parpadeo es de 1 segundo.

Un fichero *RT-Situation_Model* contiene el modelo de un modo de operación de la aplicación en el que hay formulados requisitos de tiempo real. En él se definen los modelos de la plataforma que ejecuta la aplicación, de las instancias de los módulos software que participan en ella, y de la carga de trabajo que conlleva y que se describe como conjuntos de transacciones. El aspecto mas característico de una *RT-Situation_Model* es que sólo contiene *RT-Model_Instances* y define unívocamente un modelo analizable del que se puede obtener información temporal concreta.

Dentro de la estrategia que se propone, el *RT-Situation_Model* es un fichero con estructura muy simple, que refleja de forma escueta el modelo de

Tabla 1: Ejemplo de RT_Component_Descriptor.

```
<?xml version="1.0" encoding="UTF-8"?>
<?mast fileType="CBSE-Mast-Component-Descriptor-File"?>
<MAST_COMPONENTS
   …
   xsi:schemaLocation="
   http://mast.unican.es/cbsemast/mastcomponent
      Mast_Component.xsd">
   <!--***fileName="ADQ_9111"***→
 <Component name="ADQ_9111"
    dependency="DECLARED">
    <TimeInterval name="BlinkingTime" value="1.0"/>
    <PriorityInheritanceResource name="DigMutex"
         dependency="AGGREGATED"/>
    <SimpleOperation  name="DigRead"
         wcet="1.4E+06" acet="11.25E+06" bcet="1.0E+06">
      <SharedResources>DigMutex</SharedResources>
    </SimpleOperation>
    <SimpleOperation  name="DigWrite"
         wcet="2.2E+06" acet="2.0E+06" bcet="2.0E+06">
      <SharedResources>DigMutex</SharedResources>
    </SimpleOperation>
    <SimpleOperation  name="SetBlinking"
         wcet="1.7E+06" acet="1.7E+06" bcet="1.7E+06"/>
    <RegularSchedulingServer name="BlinkingServer"
         dependency="AGGREGATED"
         scheduler="@Host">
      <FixedPriorityPolicy priority="@PollPrty"/>
    </RegularSchedulingServer>
    <Transaction name="BlinkingControl"
          dependency="AGGREGATED">
     <PeriodicExternalEvent name="BlinkingStart"
           period="BlinkingTime"/>
      <Activity inputEvent="BlinkingStart"
            outputEvent="StateON"
            activityUsage="DigWrite"
            activityServer="BlinkingSever"/>
     <Delay inputEvent="StateON" outputEvent="StateCH"
            delayMaxInterval="#@BlinkingTime/2.0#"
            delayMinInterval="#@BlinkingTime/2#"/>
      <SystemTimedActivity inputEvent="StateCH"
            outputEvent="StateOFF"
            activityUsage="DigWrite"
            activityServer="BlinkingServer"/>
      </Transaction>
      …
  </Component>
</MAST_COMPONENTS>
```

Tabla 2: Ejemplo de RT_Situation_Model.

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<?mast fileType="CBSE-Mast-Component-Descriptor-File"
version="1.0"?>
<MAST_RT_SITUATIONS
  ...
  xsi:schemaLocation="http://mast.unican.es/cbsemast/mastrts
  Mast_RT_Situation.xsd"
  fileName="Risk Analysis"
  domain="Telecontroller"
  author="Jose M. Drake Moyano"
  date="2004-06-06T00:00:00" >
  <MastRTSituation name="CarAlarm_Application"
  date="2005-03-09T00:00:00">
    <PlatformModel>
    ...
    <PrimaryScheduler name="Controller_Scheduler"...
    ...
    </PlatformModel>
    <LogicalModel>
      <ExternalComponent name="DigIO_Card"
        uri="C:\CBSE\ADQ_9111.xml"/>
      ...
      <Component name="DriverCard"
          referenced="ADQ_9111"
          dependency="AGGREGATED">
        <AssignedParameters>
            <Scheduler name="Host"
                      value="Controller_Scheduler"/>
            <Priority name="PollPrty" value="20"/>
        </AssignedParameters>
      </Component>
      ...
    </LogicalModel>
    <Transaction  ...

    ...
  </MastRTSituation>
</MAST_RT_SITUATIONS>
```

tiempo real de la aplicación, mientras que son los *RT-Model_Descriptor* que se referencian los que contienen los detalles cuantitativos y semánticos. En la tabla 2 se muestra como ejemplo una sección del *RT-Situation_Model* que describe el modelo del modo de operación "CarAlarm_ Application". En ella, se muestra la declaración de una instancia del componente "DriverCard", el cual se define a partir del descriptor de modelo "ADQ_9111" que en el modelo se indica que está localizado en la URI "C:\CBSE\ADQ_9111.xml", y al que se asigna la instancia "Controller_ Sheduler", definida también en el modelo, al parámetro "Host", y la prioridad concreta "22" al parámetro "PollPrty" del descriptor.

Con la metodología de formulación del modelo de tiempo real que se propone, el modelo de una situación de tiempo real analizable se encuentra distribuida en múltiples ficheros, lo cual hace que su estructura se adapte a la arquitectura de la aplicación, y que los modelos de los componentes sean reusables. Sin embargo, la estructura que resulta no es adecuada para ser procesada directamente por las herramientas. Como se muestra en la figura 5, el análisis de los modelos que resultan requiere una herramienta de preprocesado que hemos denominado *RT_Model_Compiler*, que procesa los modelos dispersos por el catálogo de componentes y genera un modelo compacto basado en un único fichero formulado de acuerdo con el entorno de herramientas que se esté utilizando.

En la implementación de la metodología que se ha realizado, los descriptores de modelos, las instancias de modelos y el propio modelo definido en el entorno, tienen todos ellos un formato XML común. Las diferencias entre ellos son solo de completitud de la información que contienen:

- La estructura XML de un *RT-Model_Descriptor* se define en el modelo schema "http://mast.unican.es/cbsemast/mastcomponent", y la característica de su naturaleza es que representa modelos parametrizados, y cualquier atributo o referencia puede ser expresada en función de símbolos. Todos los símbolos utilizados en la formulación del modelo que tienen un identificador con prefijo @ son parámetros que quedan indefinidos en el descriptor.
- La estructura XML de un *RT-Model_Instance* se define en el modelo W3C-Schema "http://mast.unican.es/cbsemast/mastrts". Está contenida en un único fichero que codifica un *RT-Situation_Model*. En la declaración de las instancias de cualquiera de las primitivas de modelado que contiene, se hace referencia a descriptores, e incluye una sección característica denominada "*Assigned Parameters*", en la que se asignan los valores concretos a los parámetros declarados en el descriptor que le sirve de base.
- La estructura XML de un *RT-Model* se define en el modelo W3C-Schema "http://mast.unican.es/xmlmast/model". Está contenida en un único fichero y no utiliza ningún tipo de parametrización. Se genera a partir de un *RT-Model_Instance* sustituyendo todos los parámetros por sus valores concretos.

La herramienta *RT_Model_Compiler* que se ha desarrollado, se basa en la interpretación de los diferentes tipos de modelos mediante una

herramienta estándar W3C-DOM que genera una estructura tipo árbol de cada modelo, y sustituye los símbolos por los valores concretos o los elementos a los que se hace referencia.

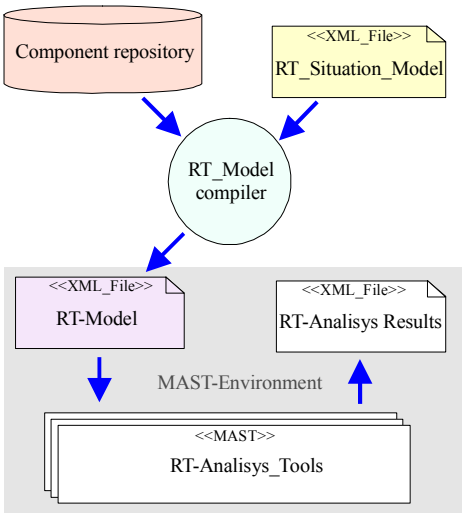

Figura 5: Herramienta de generación del modelo.

4. Conclusiones

Con la metodología de formulación del modelo de tiempo real que se propone, se simplifica el modelado de aplicaciones de tiempo real basadas en componentes ya que permite asociar modelos de comportamiento temporal reusables a los componentes. Así mismo, se facilita el modelado de sistemas de tiempo real distribuidos, en los que se debe evaluar el comportamiento de las aplicaciones en plataformas con diferentes configuraciones, para las que también esta metodología proporciona modelos reusables.

Esta metodología se ha utilizado para modelar diferentes sistemas de tiempo real distribuidos complejos, reutilizando los modelos de plataformas de ejecución (sistemas operativos, protocolos de comunicación, servicios de intermediación etc.) y de componentes software (librerías de funciones, servidores remotos, drivers, etc.). Partiendo de ella, se están desarrollando herramientas que simplifiquen el diseño y la validación de sistemas de tiempo real basadas en componentes estándar.

Referencias

[1] 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.

[2] Jane W.S. Liu: “Real-Time Systems”. ISBN 0-13-099651-3. Prentice Hall Inc., 2000.

[3] J. Stankovic. “VEST: a toolset for constructing and analyzing component based operating systems for embedded and real-time systems”. In Proc. of the Embedded Software, First International Workshop (EMSOFT 2001), volume 2211 of Lecture Notes in Computer Science, pages 390–402, Tahoe City, CA, USA, October 2001. Springer-Verlag.

[4] A. Tesanovic, D. Nyström, J. Hansson, and C. Norström: “Aspects and Components in Real-Time System Development: Towards Reconfigurable and Reusable Software”. J. of Embedded Computing, February, 2004.

[5] H. Gomaa. “Designing Concurrent, Distributed, and Real-Time Applications with UML”. Addison-Wesley, 2000.

[6] H. Kopetz, R. Zainlinger, G. Fohler, H. Kantz, P. Puschner, and W. Schütz. “The design of real-time systems: from specification to implementation and verification”. Software Engineering Journal, 6(3):72–82, 1991.

[7] A. Burns and A.Wellings. “HRT-HOOD: a Structured Design Method for Hard Real-Time Ada Systems”, volume 3 of Real-Time Safety Critical Systems. Elsevier, 1995.

[8] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M.Drake: “MAST: Modeling and Analysis Suite for Real-Time Applications” Proc. of the Euromicro Conference on Real-Time Systems, June 2001.

[9] J.L. Medina, M.González Harbour, J.M. Drake: “MAST Real-time View: A Graphic UML Tool for Modeling Object_Oriented Real_Time Systems”, RTSS, December, 2001.

[10] Object Management Group: “UML Profile for Schedulability, Performance and Time Specification”, Version 1.0. OMG document formal/03-09-01, September, 2003.