ANÁLISIS DE LA APLICACIÓN DE LA ESPECIFICACIÓN DE DESPLIEGUE Y CONFIGURACIÓN DEL OMG A SISTEMAS DE TIEMPO REAL BASADOS EN COMPONENTES.

Por Patricia López Martínez (lopezpa@unican.es)
Julio L. Medina Pasaje (medinajl@unican.es)
José M. Drake Moyano (drakej@unican.es)
Grupo de Computadores y Tiempo Real. Universidad de Cantabria

Resumen

Se analiza la especificación relativa al despliegue y configuración (D&C) de aplicaciones distribuidas basadas en componentes propuesta por el Object Management Group (OMG) [1] con el fin de su extensión a aplicaciones de tiempo real. Se utiliza la estructura del proceso de desarrollo que se propone en esta especificación, como base para analizar y definir un modelo de la información, de las tareas y de las herramientas adicionales que se requieren cuando las aplicaciones que se desarrollan son de tiempo real. El análisis se completa con el proceso de desarrollo de una aplicación de ejemplo sencilla que permite ilustrar con datos y herramientas concretas el proceso que a realizar.

I Especificación de Despliegue y Configuración de Aplicaciones Distribuidas Basadas en Componentes.

La organización OMG mantiene un grupo de trabajo que elabora la propuesta "*Deployment and Configuration of Component-Based Distributed Applications Specification*"[1] cuyo objetivo es la especificación de la información reflectiva ("*metadata*"), de las interfaces y de las herramientas que se necesitan para el despliegue y la configuración de aplicaciones basadas en componentes en plataformas distribuidas heterogéneas. Esta especificación incluye:

- La información reflectiva que describe los componentes y las aplicaciones basadas en componentes y los requerimientos que establecen a fin de ser utilizados ("*Component Data Model*") y la funcionalidad de las interfaces que almacenan, localizan y presentan esta información reflectiva ("*Component Management Model*")
- La información reflectiva que describe las plataformas distribuidas heterogéneas y su capacidad ("*Target Data Model*"), así como la funcionalidad de las interfaces que obtienen y presentan esta información ("*Target Management Model*").
- La información reflectiva que describe y especifica el despliegue de una aplicación en un plataforma ("*Execution Data Model*") y la funcionalidad de las interfaces que ejecutan el despliegue ("*Execution Management Model*").
- El proceso de despliegue que incluye la instalación, configuración, planificación, preparación y lanzamiento de la aplicación distribuida.
- Una especialización de esta información reflectiva y de las interfaces a plataformas, componentes y aplicaciones CORBA.

Su objetivo principal es constituir un estándar que facilite la compatibilidad e interoperatividad de los componentes software como piezas constructivas, de los recursos y servicios que deben proporcionar las plataformas de este tipo de aplicaciones, y de las herramientas que soporta su ciclo de desarrollo.

En la especificación D&C se utiliza el concepto de componente que ya OMG ha establecido en la especificación del lenguage UML 2.0 [2]. Un componente representa un módulo software que encapsula su información en un elemento que puede ser reemplazado dentro del entorno para el que ha sido diseñado. Un componente tiene una estructura interna opaca y define su funcionalidad y comportamiento en función de las interfaces que provee y requiere. Su objetivo es hacer posible que grandes elementos de la funcionalidad de las aplicaciones puedan diseñarse ensamblando componentes reusables, disponibles en catálogos o en otras fuentes de distribución.

La capacidad de recursividad juega un papel relevante en la definición de los componentes. De acuerdo con ella, cualquier conjunto de componentes ensamblados puede ser especificado como un componente en sí, y por lo tanto, podrá ser reusado como tal para el desarrollo de futuras aplicaciones. Dentro del contexto de esta especificación, una aplicación no es nada especial, sino sólo un componente más, cuya ejecución individual es útil. Así, un componente puede estar implementado directamente (implementación monolítica) o implementado por ensamblado de otros componentes.

A efecto de despliegue, un componente es una unidad de empaquetamiento ("*package*") que contiene tanto elementos de información reflectiva ("*metadata*"), como módulos de código compilados. La información reflectiva proporciona al diseñador la información que necesita para instalar operativamente el componente en un nodo de la plataforma, y para enlazar el componente con otros componentes que hacen uso de los servicios que ofrece en el contexto de una aplicación. Los módulos de código compilado implementan la funcionalidad comprometida a través de las interfaces especificadas en su especificación. En el caso de sistemas heterogéneos, un componente puede ofrecer múltiples y diferentes módulos de código en función de que pueda ser instanciado en diferentes tipos de nodos. Por ejemplo, un componente puede contener implementaciones alternativas para entornos Windows, Linux y Java.

A fin de instanciar o desplegar una aplicación basada en componentes, cada instancia de componente que integre, debe ser transferida e instalada en el nodo de la plataforma que tenga asignado y posteriormente debe ser instanciada, enlazada y ejecutada. Para todo ello se requieren recursos y herramientas que, conducidos por la información reflectiva, gestionen los módulos de código en cada una de las fases de despliegue.

El modelo de plataforma ("*domain*") que soporta la especificación D&C está compuesto de nodos ("*nodes*"), canales de comunicación ("*interconnect*"), redes y puentes ("*bridges*"). Los nodos tienen la capacidad computacional y constituyen el recurso que ejecuta el código de los componentes y con ello la aplicación. Esta definición incluye cualquier tipo procesador ya sea computador personal o procesador embarcado (SMPs, DSPs o FPGAs). Los canales de comunicación representan las vías de comunicación directas entre nodos que permiten el intercambio de información, como puede ser una línea ethernet o un bus CAN. Por último, un puente permite la conectividad entre diferentes canales de comunicación y representa tanto encaminadores ("*routers*") como conmutadores ("*switches*").

La plataforma de ejecución requiere estar documentada con una información reflectiva que defina las características, los recursos y las capacidades que ofrece cada elemento de que se constituye, así como las que se derivan de su interconexión. Un aspecto importante de la especificación de la plataforma, es que aunque en cada nodo se soporte un tipo específico de código de componente, su gestión durante el despliegue debe estar unificada y por tanto todos los elementos de la plataforma deben tener un modelo de descripción común, que es parte relevante de la especificación D&C.

El modelo en la especificación D&C se basa en la definición de un proceso de despliegue que cubre las fases que van desde la adquisición del componente como una unidad entregable (*"package"*) hasta su ejecución en la plataforma distribuida elegida. Todo el proceso de despliegue debe realizarse utilizando el código opaco que incluye el componente de acuerdo con la interpretación que se hace de la información reflectiva que lo acompaña. El modelo de especificación propuesto se basa en las siguientes fases:

Fase de Instalación (*"Installation"*): Se define como la tarea de adquirir componentes en un distribuidor y almacenarlo en el registro ("*repository*") del sistema de desarrollo bajo el control del diseñador. Este proceso incluye las tareas de autentificación, compatibilidad y completitud de los componentes que se adquieren, así como su formateo al estándar que se utiliza.

Fase de Configuración (*"Configuration"*): Incluye las tareas con las que se construye la aplicación desde el punto de vista de su especificación funcional, a fin de su posterior ejecución.

Fase de Planificación (*"Planning"*): Se compone de las tareas que establecen como y donde se van a ejecutar los elementos de la aplicación en el entorno de ejecución. Es la fase en la que se toman en consideración los requisitos de la aplicación con las capacidades y recursos de la plataforma y se decide que componentes y en que elementos de la plataforma se van a ejecutar.

Fase de Preparación (*"Preparation"*): Se compone de las tareas de distribución y de verificación que garantizan la disponibilidad en la plataforma de ejecución de los recursos que se van a necesitar durante la ejecución de la aplicación.

Fase de Lanzamiento (*"Launch"*): Hace referencia a las tareas que deben realizarse para ejecutar la aplicación en la plataforma de ejecución. Esto supone instanciar el código de los componentes

en los nodos asignados, establecer los enlaces entre ellos, reservar los recursos que se requieren en los nodos y en las redes de comunicación e iniciar la propia ejecución del código.

La especificación D&C es conforme con la especificación MDA ("*Model Driven Architecture*") definida también por el OMG y se ha formulado teniendo en consideración estos cuatro niveles:

- Especificación D&C para modelos independientes de la plataforma (PIM), que constituye el núcleo más relevante de la especificación.
- Especificación D&C UML que enriquece la legibilidad y comprensibilidad de la formulación PIM a fin de facilitar la transición PIM a PSM.
- Especificación D&C para modelos específicos de plataformas concretas (PSM). En la propia especificación se formula la relativa a la plataforma CCM basada en CORBA.
- Especificación D&C orientada a las herramientas. Es un perfil muy relacionado al D&C PIM y define las sintaxis abstractas y los lenguajes que requiere el proceso de distribución y configuración.

II Análisis de la extensión de la especificación D&C a aplicaciones de tiempo real.

El proceso de despliegue y configuración de las aplicaciones de tiempo real distribuidas añade a las tareas que son necesarias en el desarrollo de una aplicación distribuida convencional, otras nuevas tareas relativas a asegurar su planificabilidad y a prever que se satisfacen los requerimientos temporales establecidos en su especificación. Por tanto la extensión de este proceso consiste básicamente en añadir nuevos tipos de elementos de información reflectiva, nuevas tareas que deben realizarse en cada fase del proceso, y nuevas herramientas que se necesitan para gestionar la información y realizar los análisis asociados a las tareas.

Cualitativamente la extensión para soportar sistemas de tiempo real incrementa la complejidad del proceso de D&C, por dos razones:

- En el diseño convencional, la principal característica de un componente es su funcionalidad, y ésta se describe utilizando el concepto de interfaz. Tanto los servicios que ofrece un componente, como los que requiere para operar, se formulan mediante la enumeración de las interfaces que el componente implementa o que son requeridas a los componentes de que se sirve. En el caso de sistemas de tiempo real la caracterización de un componente necesita un nivel más bajo de descripción. El comportamiento temporal no puede formularse de forma independiente para cada servicio ofertado, sino que su formulación ha de realizarse en función de las características temporales de otros servicios ofertados por otros componentes externos a él y de las capacidades de los recursos de la plataforma que utiliza. Esto supone que la descripción opaca de un componente requiere de una compleja información que abstrae la descripción de los servicios ofertados en función de la descripción de los servicios requeridos, que denominamos modelo de tiempo real del componente.

- El diseño convencional requiere básicamente un análisis de compatibilidad entre las especificaciones y los requerimientos de los diferentes componentes entre sí y las de los recursos de la plataforma. Estos pueden ser formulados mediante información de naturaleza enumerada, con lo que las tareas de D&C se reducen a sencillas operaciones de comparación de conjuntos. Por el contrario en el diseño de aplicaciones de tiempo real el diseño requiere el procesamiento conjunto de las especificaciones de los componentes y de la plataforma para verificar que satisfacen los requerimientos temporales previstos en la aplicación. En este caso, gran parte de la información que se gestiona es de naturaleza continua (magnitudes reales) y la evaluación de sus interacciones requiere un procesamiento algebraico y numérico mucho más complejo.

El proceso de diseño de aplicaciones basada en componentes tiene en común con el diseño de tiempo real la existencias de dos etapas. Una en la que se diseñan funcionalmente los módulos de la aplicación, y otra en la que se analiza la componibilidad de los módulos y la funcionalidad resultantes de su composición. Aunque los conceptos de componibilidad y validación son de naturaleza diferente en el análisis funcional y en el análisis y diseño de tiempo real, el proceso que soporta ambos es compatible. Si hacemos un análisis de las fases del proceso de D&C que incluye la propuesta OMG y consideramos su naturaleza y objetivo, se comprueba que se adapta muy bien al diseño de aplicaciones de tiempo real, siempre que

incorporemos nuevos tipos de datos para describir los aspectos que le son propios, e identifiquemos las tareas que requiere el diseño de tiempo real.

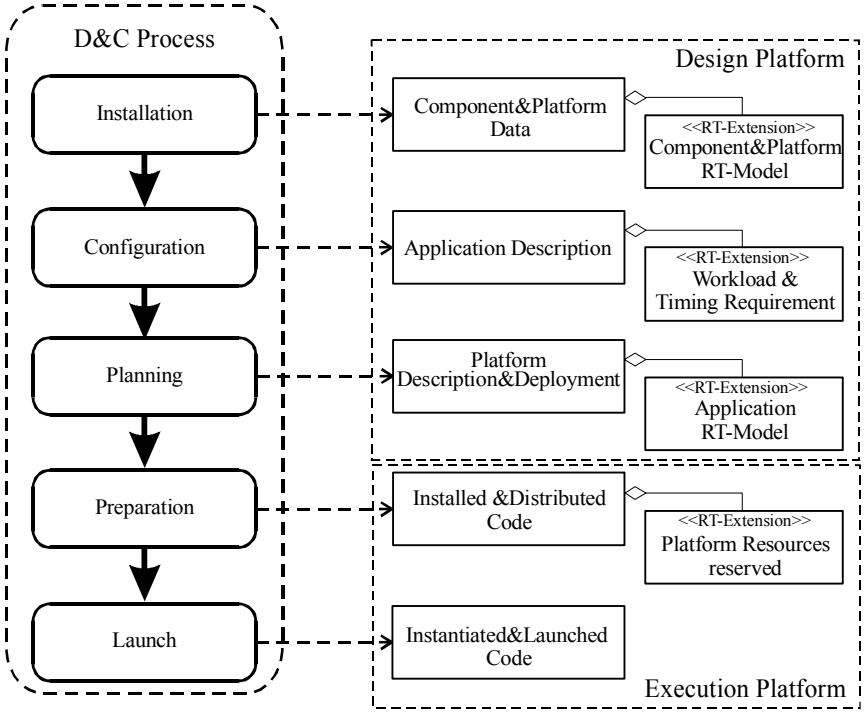

Figura 1.- Extensión del proceso D&C a aplicaciones de tiempo real.

Fase de Instalación: El objetivo de esta fase es reunir en la plataforma de diseño la información relativa a los componentes disponibles para el diseño y de los elementos con los que se puedan construir posibles plataformas de ejecución.

Extensión de tiempo real: La extensión de la especificación D&C para aplicaciones de tiempo real requiere la definición de los nuevos tipos de información que se necesitan para describir las características de tiempo real de:

- Modelos de los componentes software: Su modelo de tiempo real contiene la información que se necesita para describir el comportamiento temporal que se deriva de su funcionalidad, que incluye:

  - La temporización de los servicios que ofrece.
  - Los mecanismos de sincronización que utiliza para controlar de forma segura su acceso.
  - En el caso de componentes activos los modelos de las transacciones que tienen su origen en él.

  El modelo de tiempo real de un componente es un modelo parametrizado, que define los valores de características o las referencias a otros modelos que permiten particularizar el modelo a cada instancia del componente que se defina en el contexto de una aplicación. Un parámetro común a todos los componentes software es el que hace referencia al modelo del nodo en que se ejecutará cada instancia del componente. A través de él, el modelo obtiene la información sobre la capacidad de procesamiento disponible, naturaleza y características del planificador, etc., con la que las herramientas evalúan los tiempos de ejecución de los servicios a partir de los datos sobre cantidad de procesamiento requerida que se especifica en el descriptor.

- Modelos de los elementos de la plataforma: La plataforma de ejecución es el conjunto de recursos (nodos, sistema operativos, redes de comunicación, etc.) que se agrupan a fin de ejecutar la aplicación. En un sistema de tiempo real, las características y capacidades de la plataforma son esenciales para evaluar el comportamiento temporal de los componentes software que se ejecutan en ella. Actualmente las plataformas son estructuras modulares, en las que los procesadores, sistemas operativos, redes de comunicación, drivers, etc. pueden asociarse en diferentes combinaciones, por lo que es importante disponer también de modelos modulares que permitan

construir el modelo de tiempo real completo de las plataformas que puedan utilizarse. Modelos de tiempo real de recursos de la plataforma son:

- Modelos de los procesadores: Describen la capacidad de procesamiento de un nodo que se deriva del procesador y del hardware que integra: capacidad de procesamiento, granularidad de la temporización y modelo de la gestión de las interrupciones
- Modelos de los sistemas operativos: Describen las características de administración de los recursos software, y del costo en capacidad de procesamiento que supone su gestión: estrategias de planificación, capacidad de procesamiento que consumen la gestión de procesos, la gestión del tiempo, etc.
- Modelos de las redes de comunicación: Describen la anchura de banda que proporciona, el nivel de granularización en paquetes de los mensajes que transfiere y la capacidad de planificar la transferencia concurrente de mensajes, así como, el costo de anchura de banda que le requiere la sincronización de los diferentes nodos.
- Modelos de los drivers de comunicación: Describen la capacidad de procesamiento de los nodos que se consume en la atención y gestión de las redes por las que se comunican.
- Modelos de software de intermediación: Describen el comportamiento temporal y la capacidad de procesamiento que consumen los servicios de base con que están dotados los nodos, como por ejemplo, los recursos de acceso a servicios remotos (APC y RPC).

Fase de Configuración (*"Configuration"*): El objetivo de esta fase es la descripción de la aplicación como una agregación de componentes interconectados, así como la elaboración de su especificación.

Extensión de tiempo real: La definición estructural de una aplicación como agregación y enlazado de componentes predefinidos en las aplicaciones de tiempo real es similar a la de aplicaciones convencionales. Sin embargo, la descripción de una aplicación de tiempo real requiere además definir la carga de trabajo que debe planificarse y las restricciones temporales que han de satisfacerse. Por ello, en una aplicación de tiempo real deben identificarse las situaciones de tiempo real, esto es, los modos de operación para los que existen definidos requisitos temporales, y para cada una de ellas se ha de caracterizar las transacciones que pueden concurrir en su ejecución. Elementos adicionales que se requieren en la fase de configuración de una aplicación de tiempo real serán los requisitos de tiempo real y las transacciones que concurren en ella:

- Modelo de una situación de tiempo real: Describe cada modo de operación de la aplicación en los que existen definidos requerimientos temporales. La descripción de una situación de tiempo real, requiere definir la aplicación desde dos puntos de vista complementarios. Desde el punto de vista estructural, la aplicación se describe como el conjunto de instancias de componentes que participan en su ejecución, y desde el punto de vista reactivo, se describe la carga de trabajo que lleva a cabo y los requerimientos temporales que se establecen sobre su ejecución. Tradicionalmente en los sistemas de tiempo real, este comportamiento reactivo se define como conjuntos de transacciones que se invocan concurrentemente en él.

- Modelo de una transacción: Describe cada una de las secuencias de actividades que se ejecutan como respuesta a un patrón de eventos externos o de temporización. La capacidad de procesamiento que requiere la ejecución de las actividades de la transacción y el patrón temporal de generación de los eventos que las disparan, definen la carga de trabajo que genera. A su vez, el modelo de la transacción es el marco sobre el que se definen los requerimientos temporales establecidos en la especificación de la aplicación.

El modelo de tiempo real de la aplicación que resulta en la fase de configuración no es suficiente para evaluar su planificabilidad o las prestaciones temporales que ofrece, ya que aún no se ha definido la plataforma de ejecución, de la que también dependen.

Fase de Planificación (*"Planning"*): El objetivo de esta fase es establecer la distribución de la aplicación sobre la plataforma, esto supone en primer lugar, definir la plataforma identificando los elementos de que se compone y sus asociaciones, y en segundo lugar, asociar los componentes de la aplicación a los nodos en que se ejecuta.

Extensión de tiempo real: Las tareas principales en esta fase son la especificación estructural de la plataforma de ejecución como conjuntos de recursos y asociaciones que se declaran como

constituyentes de ella, así como la distribución de las instancias de los componentes software con los que se implementa la aplicación sobre los recursos de la plataforma. Estas tareas son compatibles con las que se realizan en cualquier tipo de aplicación, sea o no de tiempo real.

El aspecto específico de aplicaciones de tiempo real más importante de esta fase es que en ella queda completamente definido el modelo de tiempo real de la aplicación, y con él, el diseñador dispone de la información necesaria para analizar su planificabilidad y evaluar sus prestaciones temporales. En el caso de que la plataforma sea dedicada, el diseñador podrá certificar su planificabilidad, y si la plataforma es abierta, esto es, se encuentra ejecutando otras aplicaciones de carga desconocida, el diseñador podrá definir los requerimientos de prestación de servicios que deben ser solicitados a la plataforma para que la aplicación pueda ser ejecutada satisfaciendo su especificación temporal.

Fase de Preparación (*"Preparation"*): El objetivo de esta fase es garantizar la disponibilidad en la plataforma de ejecución de los recursos que van a necesitarse en la ejecución de la aplicación.

Extensión de tiempo real: La tarea básica de esta fase consiste en verificar la disponibilidad de los recursos en los nodos de la plataforma que se requieren para la ejecución de la aplicación, y en su caso transferir a ellos los que no estén disponibles.

En el caso de sistemas de tiempo real esta fase es el marco adecuado para la negociación y reserva de los servicios cuya prestación debe ser garantizada por los diferentes recursos de la plataforma, a fin de que la aplicación pueda ser ejecutada satisfaciendo los requerimientos temporales.

Fase de Lanzamiento (*"Launch"*): El objetivo de esta fase es ejecutar la aplicación en la plataforma de ejecución.

Extensión de tiempo real: En esta fase, las instancias previstas de los componentes son creadas en los nodos, los enlaces entre ellos y con los recursos de la plataforma son establecidos y se inicia su ejecución. Estas tareas son similares a las establecidas para las aplicaciones convencionales.

III Ejemplo de desarrollo de una aplicación de tiempo real basada en componentes.

A fin de hacer más descriptiva y concreta la información que se requiere para el despliegue y configuración de una aplicación distribuida basada en componentes, se propone el siguiente ejemplo sencillo, y se analizan las tareas que se realizan y la información que se gestiona en cada fase. En este ejemplo se utiliza la metodología de modelado de sistemas de tiempo real CBSE-Mast [3].

Se propone una aplicación de tiempo real cuya función es chequear periódicamente el estado de un conjunto de líneas físicas digitales con diferentes periodos, y en caso de que su estado no sea el definido como correcto, ejecutar acciones de alarma consistentes en generar sonidos y activar ciertas líneas digitales de salida. En la figura 2, se muestra el modelo lógico de diseño de la aplicación, que utiliza tres tipos de componentes software:

Componente Agent: Es un componente de tipo cliente activo que controla la ejecución concurrente de múltiples tareas de verificación de alarma utilizando un thread independiente para cada una de ellas. Al ser un componente cliente su contrato de uso no ofrece ninguna interfaz. En su contrato de instanciación se establece que requiere tres tipos de servidores: bajo el role *sensor* puede requerir un conjunto de componentes que han de ofrecer la interfaz I_Digital y de los que lee el estado de las líneas de entrada que verifica. Bajo el role *actuator* puede tener enlazados un conjunto de componentes que deben ofrecer también la interfaz I_Digital, para con ella ejecutar acciones de alarma. Por último, bajo el role *speaker*, puede tener enlazado un componente opcional que debe ofrecer la interfaz I_Player, para con él ejecutar los sonidos de alarma.

Componente IO_Card: Es un componente pasivo y servidor que en su contrato de uso ofrece la interfaz I_Digital, a través de la que los clientes pueden leer una señal digital mediante la función *readState*, o establecer una línea digital de salida a través del procedimiento *putState*. Es un servidor terminal que en su contrato de instanciación no requiere ningún otro servidor.

Componente SoundPlayer: Es un componente servidor activo que en su contrato de uso ofrece la interfaz I_Player, a través de la que un cliente puede generar sonidos de alarma invocando el procedimiento *play*. En el contrato de instanciación, requiere bajo el role *instrument* un componente que ofrezca la interfaz I_Digital para con ella actuar sobre algún dispositivo que genera los sonidos. Es activo porque en cada instancia del componente se introduce un thread que controla la reproducción temporizada de los sonidos.

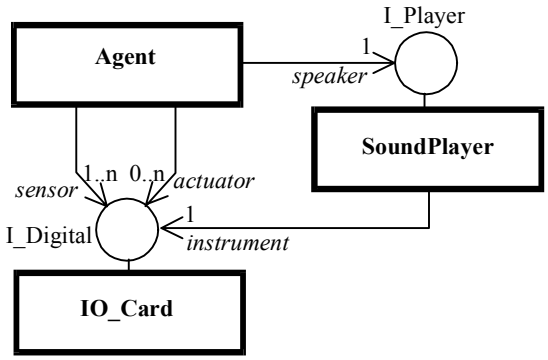

Figura 2: Arquitectura software de la aplicación ejemplo.

Fase de Instalación

La fase de instalación de esta aplicación consiste en el registro en la base de datos de la plataforma de diseño, de la información y código que constituyen los componentes utilizados en la aplicación, y la información y modelos que describe los elementos con los que se puede construir la plataforma de ejecución de la aplicación. En la figura 3, se muestra un ejemplo de los elementos que se podrían encontrar en el registro (*repository*) durante el diseño de la aplicación ejemplo.

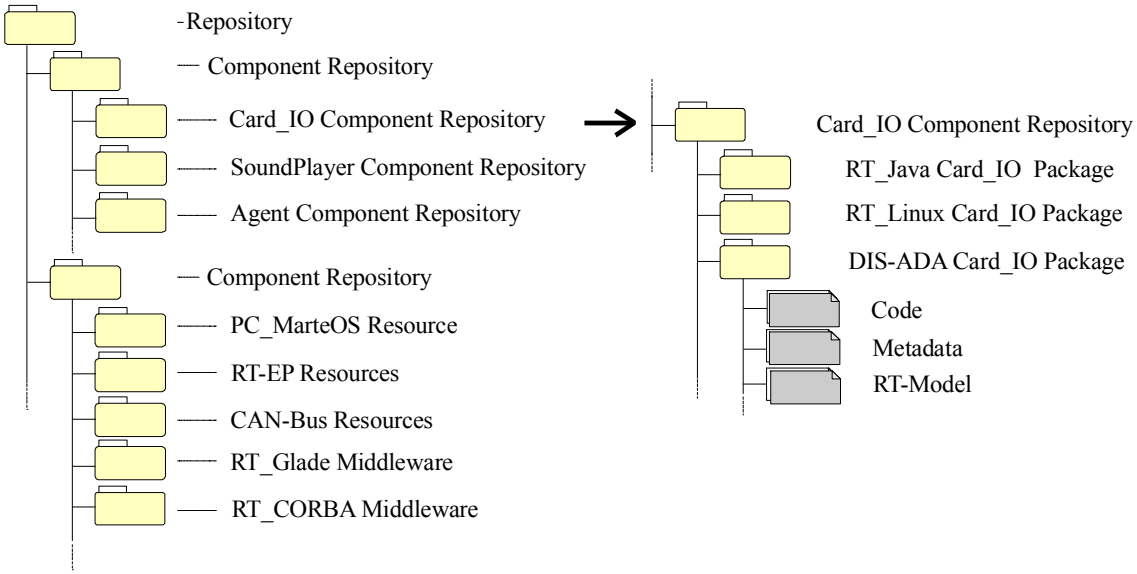

Figura 3: Estructura y elementos típicos del Registro.

Como ejemplo del modelo parametrizado que describe el comportamiento temporal de un tipo de componente, se muestra el modelo "RT_Card_IO" que describe el modelo de tiempo real del componente IO_Card cuya funcionalidad es el control de la tarjeta PCI-9991 de adquisición de señales analógicas y digitales. El modelo describe la temporización de los servicios que el componente ofrece a sus clientes. En la sección de modelo que se muestra en la Tabla 1, se ve la declaración del modelo temporal de la operación *readState*, a través de la cual se puede leer una línea digital de entrada. El modelo incluye tanto las características temporales que implica la ejecución local del código que implementa la operación

<<Simple_Operation>> localReadState, como los parámetros adicionales que se requieren para caracterizar el mismo servicio cuando se invoca remotamente <<RPC_Operation>> readState.

Tabla 1.- Sección del modelo de tiempo real del componente IO_Card

```
<?xml version="1.0" encoding="UTF-8"?>
<?mast fileType="CBSE-Mast-Component-Descriptor-File" version="1.1"?>
<!-- Real-time model of the logical component "ADQ_9111" -->
<mast_c:MAST_COMPONENTS
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:mast_c="http://mast.unican.es/cbsemast/component"
      xmlns:mast="http://mast.unican.es/cbsemast/mast"
      xmlns:mast_u="http://mast.unican.es/cbsemast/umlmast"
      xsi:schemaLocation="http://mast.unican.es/cbsemast/component c:\…\Mast_Component_v2.xsd"
      fileName="RT_Card_IO" domain="ADQ" author="Patricia Lopez" version="2006-01-11">

   <mast_c:Component name="RT_Card_IO" roles="DIGITAL_IO ANALOG_IO">
      <mast_c:Info> Real-time model of the logical component "C_ADQ_9111"</mast_c:Info>

      <mast_c:ParamInfo name="@theHost" type="ComponentRef"
            info="The host that executes the software Component"/>
      <mast_c:ComponentRef name="host"  value="@theHost" reqRoles="RCI_PROC_NODE"/>

      <!-- Models of  remote services offered by the component-->

      <mast_c:SimpleOperation  name="localReadState" wcet="2.8E+06" acet="2.8E+06" bcet="2.8E+06"/>

      <mast_c:RPCOperation name="ReadState" info="Model of remote digital line reading service invocation"
            usage="localReadState">
         <mast_u:OutgoingMessage minMessageSize="32" maxMessageSize="32" avgMessageSize="32"/>
         <mast_u:OutgoingMarshalling acet="6.5E-06" bcet="6.1E-06" wcet="6.0E-06"/>
         <mast_u:OutgoingUnmarshalling acet="4.7E-06" bcet="4.4E-06" wcet="4.4E-06"/>
         <mast_u:IncomingMessage minMessageSize="32" maxMessageSize="32" avgMessageSize="32"/>
         <mast_u:IncomingMarshalling acet="3.5E-06" bcet="3.3E-06" wcet="3.3E-06"/>
         <mast_u:IncomingUnmarshalling acet="1.8E-06" bcet="1.8E-06" wcet="1.8E-06"/>
      </mast_c:RPCOperation>

      <mast_c:SimpleOperation  name="localPutState" …
      …..
      <!-- Model of common element declared in the component -->
      <mast_c:ParamInfo name="@theMutexCeiling" type="Priority"
            info="Techo de prioridad del mutex asociado al recurso"/>
      <mast_c:ImmediateCeilingResource name="mutex" ceiling="@theMutexCeiling" preassigned="NO"
            tie="AGGREGATED" info="The mutex for access to the register of the card "/>
   </mast_c:Component>
</mast_c:MAST_COMPONENTS>
```

Parámetros que están definidos en el modelo de este componente, y que deben ser establecidos para cada instancia del componente que se necesite modelar, son:

@theHost	Que referencia el modelo de tiempo real del nodo de procesamiento en el que se ejecuta la instancia.
@theMutexCeiling	Que permite particularizar para cada instancia el techo de prioridad del mutex que controla el acceso con exclusión mutua al registro hardware que controla las salidas digitales.

Obsérvese que el modelo incluye tanto la información cuantitativa que describe el comportamiento temporal del componente como la información reflectiva que describe su semántica (atributos *info* y elementos <*info*> y <*ParamInfo*> .

Fase de Configuración

En esta fase se diseña la aplicación como composición de instancias de componentes, y en el caso de aplicaciones de tiempo real, se formula el modelo de comportamiento de la aplicación. En la figura 4 se muestra la aplicación *CarAlarm* que se diseña en este ejemplo. Se compone de una instancia del componente tipo Agent, que tiene como servidores un componente de la clase SoundPlayer enlazado con el role *speaker* y tres componentes de la clase IO_Card enlazados con los roles *sensor* y *actuator*. La

configuración de la aplicación desde el punto de vista de tiempo real requiere la elaboración de dos tipos de información:

- El modelo de tiempo real de la aplicación se construye asociando un elemento de modelo por cada instancia de componente definida en la aplicación. Estos elementos se declaran, haciendo referencia al modelo genérico del componente del que son instancia definido en el Registro, y especificando los valores concretos que se asignan a los parámetros definidos en él, de acuerdo con la naturaleza o situación de la instancia del componente dentro de la aplicación. En la tabla 2 se muestra una sección de este modelo en el que se declaran las instancias *boardSpeaker* y *boardPanel* que son instancias de componentes del tipo SoundPlayer e IO_Card respectivamente, y se establecen los valores de los parámetros que corresponden a las instancias en esta aplicación.

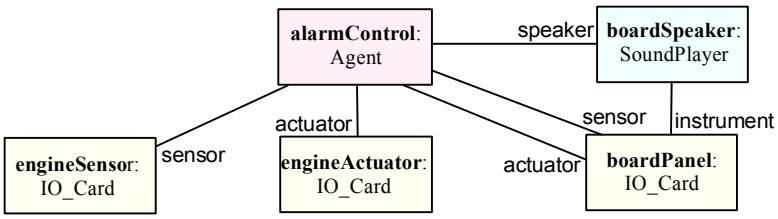

Figura 4: Configuración de la aplicación por composición de componentes.

Tabla 2. Sección de la declaración del modelo de la aplicación CarAlarm.

```
<mast_i:LogicalModel>
….
    <mast_c:ExternalComponent name="Card_mdl" base="RT_IO_Card" tie="DECLARED"
            uri="c:\...\Components\RT_IO_Card.xml"/>

    <mast_c:Component name="boardPanel"  base="LOGICAL.Card_mdl" tie="AGGREGATED">
       <mast_c:AssignedParameters>
            <mast_c:ComponentRef name="theHost"  value="@UNRESOLVED"/>
            <mast_c:Priority name="theMutexCeiling" value="20"/>
       </mast_c:AssignedParameters>
    </mast_c:Component>

    <mast_c:ExternalComponent name="Speaker_mdl" base="RT_Speaker" tie="DECLARED"
            uri="c:\...\Components\RT_Speaker.xml"/>

    <mast_c:Component name="boardSpeaker" base="LOGICAL.Speaker_mdl" tie="AGGREGATED">
        <mast_c:AssignedParameters>
            <mast_c:ComponentRef name="theHost"  value="@UNRESOLVED"/>
            <mast_c:ComponentRef name="theCommNetwork" value="@UNRESOLVED"/>
            <mast_c:Priority name="theSoundPriority" value="5"/>
            <mast_c:ComponentRef  name="theInstrument" value="LOGICAL.boardPanel"/>
            <mast_c:ComponentRef name="theInstrumentAccess" value="@UNRESOLVED"/>
        </mast_c:AssignedParameters>
    </mast_c:Component>
…
</mast_i:LogicalModel>
```

- Se define la carga de trabajo de la aplicación y los requerimientos temporales que deben satisfacer. En la tabla 3 se muestra la declaración de la transacción *oilPressureControl,* que es declarada en el modelo del componente activo *Agent* ya que es gestionada por él. Su funcionalidad se describe en el diagrama de secuencias de la figura 5.

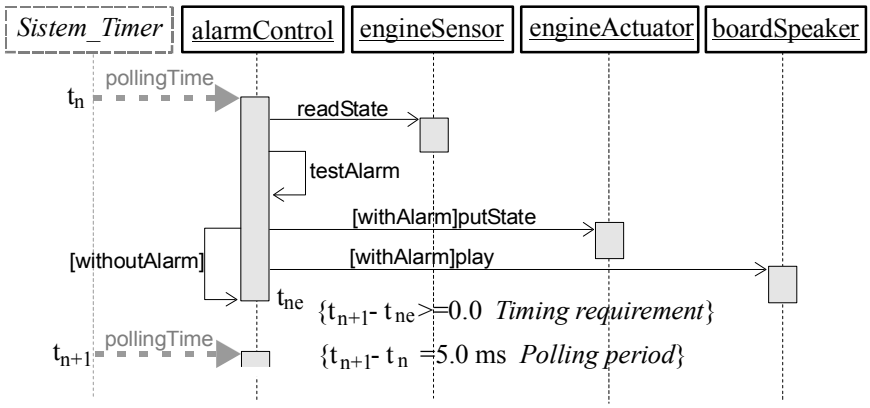

Figura 5: oilAlarmTrans (a typical alarm transaction).

Tabla 3: Declaración de la transacción OilPressureControl

```
<mast_i:LogicalModel>
…
  <mast_c:ExternalComponent name="Agent_mdl"  base="RT_Agent" tie="DECLARED"
          uri="c:\...\Components\RT_Agent.xml"/>
  <mast_c:Component name="alarmControl" base="LOGICAL.Agent_Model" tie="AGGREGATED">
  …
    <mast_c:Transaction name="OilPressureControl" base="ControlAlarmTask" tie="AGGREGATED">
        <mast_u:AssignedParameters>
            <mast_u:Priority name="controlAlarmPriority" value="20"/>
            <mast_u:TimeInterval name="controlAlarmPeriod" value="5.0E-3"/>
            <mast_u:ComponentRef name="usedSensor" value="engineSensor"/>
            <mast_u:ComponentRef name="usedActuator" value="engineActuator"/>
            <mast_u:Positive name="checkPerAlarm" value="990"/>
        </mast_u:AssignedParameters>
    </mast_c:Transaction>
  …
  </mast_c:Component>
…
</mast_i:LogicalModel>
```

Fase de Planificación

En esta fase del proceso de despliegue, se declara la plataforma en la que se va a ejecutar la aplicación y se realiza la distribución de las instancias de los componentes en los nodos de la plataforma. En la figura 6 se muestra el diagrama de despliegue que describe la plataforma de ejecución que se va a utilizar en la aplicación ejemplo *carAlarm*, y la distribución de sus componentes en ella. La plataforma está compuesta de dos procesadores denominados *panelProcessor* y *engineProcessor*, que se comunican entre sí a través del bus de comunicaciones *localBus*. La aplicación se ha realizado utilizando ADA Distribuido y el software de comunicación se realiza utilizando el software de intermediación RT_GLADE[4]. La aplicación despliega en la plataforma cinco instancias de los componentes. Los componentes *alarmControl* de tipo Agent, el componente *boardPlayer* de tipo SoundPlayer y el componente *boardPanel* del tipo IO_Card en el procesador *panelProcessor* y los componentes *engineSensor* y *engineActuator*, ambos del tipo IO_Card en el procesador *engineProcesor*.

En esta fase, se completa el modelo de tiempo real de la aplicación, para ello debe incluirse en el modelo de la aplicación el modelo de los recursos que constituyen la plataforma, y así mismo, se asignan a los parámetros de los modelos de los componentes las referencias a los modelos de los recursos de la plataforma.

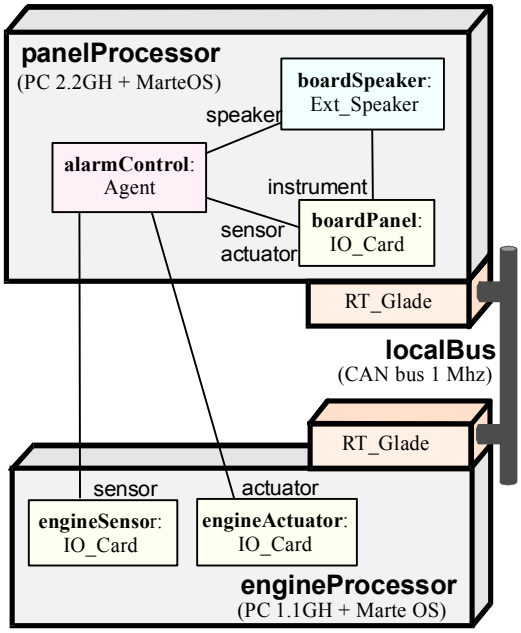

Figura 4: Arquitectura de la plataforma y despliegue de la aplicación.

En la tabla 4, se muestra una sección del modelo de tiempo real de la aplicación en la que se declara los nodos *panelProcessor* y *engineProcessor* de la plataforma, así como el enlace del modelo del componente *Agent* con el nodo de procesamiento en que se ejecuta, con la red de comunicación que se utiliza, y el software de comunicaciones con el que se invoca el componente *engineSensor*

Tabla 4. Declaración del modelo de tiempo real de la plataforma y asignación de los componentes.

```
…
<mast_i:MastRTSituation name="carAlarm_Application" date="2006-01-11T00:00:00">
    <mast_i:PlatformModel>

        <mast_c:ExternalComponent name="RCINode_mdl" base="RCI_Glade_Node" tie="DECLARED"
              uri="c:\...\Components\RT_PC_MarteOS.xml"/>
        <mast_c:Component name="panelProcessor" base="PLATFORM.RCINode_mdl" tie="AGGREGATED">
            <mast_c:AssignedParameters>
                <mast_c:Float name="theSpeedFactor" value="1.0"/>
            </mast_c:AssignedParameters>
        </mast_c:Component>
    …
    </mast_i:PlatformModel>
    <mast_i:LogicalModel>
   …
        <mast_c:ExternalComponent name="Agent_mdl"  base="RT_Agent" tie="DECLARED"
                uri="c:\...\Components\RT_Agent.xml"/>
        <mast_c:Component name="alarmControl" base="LOGICAL.Agent_Model" tie="AGGREGATED">
            <mast_c:AssignedParameters>
                <mast_c:ComponentRef name="theHost" value="PLATFORM.panelProcessor"/>
                <mast_c:ComponentRef name="theCommNetwork" value="PLATFORM.comm"/>
                <mast_c:ComponentRef name="theSpeaker" value="LOGICAL.boardSpeaker"/>
                <mast_c:ComponentRef name="theSpeakerAccess" value="PLATFORM.RPCService_Model"/>
            </mast_c:AssignedParameters>
            …
        </mast_c:Component>
       …
    </mast_i:LogicalModel>
</mast_i:MastRTSituation>
```

Una vez que ha quedado completo el modelo de tiempo real de la aplicación en esta fase, el diseñador puede utilizar las herramientas de análisis y diseño a fin de optimizar los parámetros de ejecución y verificar su planificabilidad. Por ejemplo, pueden ser evaluadas las prioridades óptimas de los entes de planificación, de los mensajes en las redes y de los recursos compartidos; de igual forma se pueden aplicar las herramientas de análisis de planificabilidad para verificar la capacidad de la plataforma para ejecutar la aplicación satisfaciendo los requerimientos temporales especificados en ella.

La estrategia de análisis del modelo de tiempo real es diferente según la plataforma de ejecución:

- Cuando la plataforma de ejecución es dedicada. En este caso, el diseñador conoce la totalidad de la carga de trabajo disponible en la plataforma de ejecución y puede realizar un análisis completo del modelo. Del análisis puede resultar la garantía de que la aplicación es planificable en la plataforma propuesta, y así mismo, estimar los tiempos de respuesta, las holguras en los requerimientos temporales, el porcentaje de utilización de los recursos, etc.
- Cuando la plataforma de ejecución es abierta, esto es, si debe ejecutar además otras aplicaciones que son desconocidas para el diseñador, el análisis del modelo de tiempo real genera como resultado los recursos y capacidades de servicios (capacidad de procesamiento en los nodos, la anchura de banda en las redes, etc.) de la plataforma que deben ser requeridos y contratados para que la aplicación sea planificable.

Fase de Preparación.

En esta fase hay que proveer la disponibilidad de los recursos de la plataforma de ejecución para poder ejecutar la aplicación. En general, en esta fase hay que verificar la disponibilidad del código de la aplicación en los nodos de la plataforma, y en el caso de que se necesite, su transferencia.

En el caso de sistemas de tiempo real que se ejecuta sobre plataformas abiertas, en esta fase se debe negociar y reservar los servicios que deben ser provistos a la aplicación.

IV Conclusiones.

El diseño de aplicaciones basada en componentes y el diseño de sistemas de tiempo real tienen como característica común el ser procesos dirigidos por modelos, y aunque la naturaleza de los modelos sea distinta y las herramientas que ayudan a la toma de decisiones de diseño sean diferentes, hay otros muchos aspectos compartidos, como son las coincidencias de fases en los procesos, la gestión de los modelos o la necesidad de información reflectiva para facilitar su gestión. El paralelismo entre ambos procesos aún es de mayor interés, si estamos considerando sistemas de tiempo real basados en componentes, que requieren la concurrencia de ambos procesos en el diseño de un mismo sistema.

El gran interés tecnológico que tiene actualmente la tecnología de componentes software, y en particular para el caso de plataformas distribuidas, ha hecho que el OMG aborde la estandarización de los procesos y de la gestión de la información de los componentes, de la configuración de las aplicaciones y del despliegue de éstas sobre plataformas distribuidas. Aunque actualmente el grupo de trabajo del OMG no parece interesado en la consideración de los aspectos no funcionales en la propuesta del estándar, el que en el estudio de estos aspectos, en los que se incluyen los relativos al diseño de sistemas de tiempo real, se trate de desarrollar la tecnología a la sombra de este estándar, compartiendo todos aquellos conceptos que son comunes y útiles, va a redundar en una mayor facilidad de avance, al encontrar un marco conceptual ya aceptado por la industria y seguir un camino que ya está formalizado.

Referencias

[1] OMG: *"Deployment and Configuration of Component-based Distributed Applications Specification"* Draft Adopted Specification ptc/03-07-02. June 2003. http://www.omg.org

[2] U2 Partners, "*Unified Modeling Language: Superstructure*," version 2.0. Recommended for adoption. April 2003. http://www.omg.org.

[3] Medina J.L.,López Martínez P. Y Drake J.M.: "Metodología de modelado de sistemas de tiempo real orientada a la componibilidad" CEDI 1er Congreso Español de Informática, Simposio STR, pp.27-34, Granada Septiembre 2005.

[4] López-Campos, J.-J. Gutiérrez and M. González-Harbour: "The chance for Ada to support distribution and real-time in embedded systems" Procc. of the 8th Intl. Conf. on Reliable Software Technologies, Palma de Mallorca, Spain , June 2004